

UNIVERSITY OF CALIFORNIA, SANTA BARBARA

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO



SANTA BARBARA • SANTA CRUZ

SANTA BARBARA, CALIFORNIA 93106

REMOTE SENSING INFORMATION SCIENCES RESEARCH GROUP

BROWSE IN THE EOS ERA

FINAL REPORT

1 MAY 1989

PRINCIPAL INVESTIGATORS
PROF. JOHN E. ESTES
DR. JEFFREY L. STAR

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D.C. 20546

GRANT NO. NASA NAGW-987

(NASA-CR-184637) REMOTE SENSING INFORMATION
SCIENCES RESEARCH GROUP: EOS IN THE EOS
ERA Final Report (California Univ.) 155 P
CSCL 08B

N89-22979

Unclas
0211724
G3/43

Browse in the EOS Era

Final Report: Year 1

Table of Contents

I.	Abstract	2
II.	Narrative	3
III.	Papers and Presentations	7
	Knowledge-Based Image Data Management: An Expert Front-End for the Browse Facility	
	Browse Capability for Spatial Databases	
	Electronic Browsing for Suitable GIS Data	
	MAPWD: An Interactive Mapping Tool for Accessing Geo-Referenced Data Sets	
IV.	Appendices	
	Meetings	
	Recent Testbed Users	
	Source Code	
	Browse User Manual	
	Image Display User Manual	

I. Abstract

The Information Sciences Research Group, University of California, Santa Barbara has undertaken a program of research and development over the past two years to examine the problem of science data browse. Given the tremendous data volumes that are planned for future space missions, particularly the Earth Observing System in the late 1990's, we as scientists must understand our needs for access to large spatial databases.

During this second contract year, we continued to work with a multi-disciplinary group of colleagues, to refine our concept of data browse. Further, we have developed software to provide a testbed of our concepts, both to locate possibly interesting data, as well as view a small portion of the data. We have placed Build II of this software testbed on a minicomputer and a PC in our laboratory, and provided accounts for our colleagues to use the testbed. A number of colleagues have begun to consider our testbed software as an element of their in-house data management plans, and we are continuing to work with and advise them.

II. Narrative

Our progress reports have detailed the background and justification for this effort. Briefly, space programs planned for the next two decades will be able to provide extraordinary volumes of digital data. The Earth Observing System program, planned for the late 1990's, is being designed to be able to produce 1 to 2 terrabits of data per day. Our work in the last contract year is designed to help us focus on the logical interface between the working scientist and the tremendous data volumes coming from these kinds of programs, which will be stored in geographically distributed repositories. We explicitly assume that network capabilities will be in place to support the kinds of data transfer that a graphics-based browse capability will require.

Our working hypothesis has been that there are user-friendly mechanisms which may help us to be able to examine reduced-volume representations of elements of a dataset (i.e., a **browse function**), which will permit us to determine the value or relevance of this data for further analysis. Two separate functions are required, in a broad sense. First, we must be able to locate collections of potentially interesting data. This kind of function is recognized in the NASA Ocean Data System, as well as at NSSDC and the US Geological Survey (as evidenced in their Earth Science Data Directory). Further, we believe that mechanisms must be in place to be able to examine some portion of a data granule (as defined by the ESADS workshop in early 1987), to be able to make a reliable decision on the utility of the data for a science problem. A good example of the latter function is the microfiche browse images which have been produced during the Landsat program.

The definition of *browse* that was developed at the Earth Sciences and Applications Data Systems Workshop in February 1987 has guided our work. The definition is:

A browse system enables a scientist to interactively subset a large number of granules or a portion of single granule for analysis. The intention of browse is not to duplicate all general software for graphics display of data, or the manipulation of images, but to provide the user with a subset of browse information to give a "feel" for the data and its coverage. Different disciplines and different data sources will probably require different functions to provide the relevant information to the users. The browse process must be "fast". For example, screen display should be presented in 30 seconds or less so that a significant number of granules can be examined in a single session. Since some circumstances require that the data upon which the browse is based be recent (i.e., quick look), long time delays between data acquisition and availability may not always be acceptable.

The overall software architecture, database design, and hardware we are using are described in the papers and presentations section of this report. Overall, the software, written largely in Fortran 77, runs on a DEC Microvax II, with the GKS library providing device-independent graphics code. Users may work with simple ASCII terminals, VT-series terminals, or Tektronix-like monochrome graphics terminals.

Browse in the EOS Era

System access is provided by 300/1200/2400/9600 BPS dial-up lines or via the SPAN and ARPANet networks. We provide a list of the off-campus users of the system at the end of this report.

In addition to the scientists who have joined us at the BROWSE meetings we have sponsored, we are attempting to stay in contact with a number of other individuals. This include:

Dr. Cecil Bloch
Research Libraries Group, Jordan Quad, Stanford, CA 94350
Dr. Harvey Croze, United Nations Environment Program
PO Box 30552, Nairobi, Kenya
Dr. Sarah Graves, Computer Science Department
University of Alabama, Huntsville
Dr. George Ludwig
Univ. Colo., Campus Box 10, LASP, Boulder, CO 80309
Mr. Wayne Moonyhan, United Nations Environment Program
6, Rue del la Gabelle, CH 1227 Carouge, Geneva, Switzerland
Mr. Jim Weber
Public Service Satellite Consortium, Washington, D.C.
Ms. Denise Wiltshire, USGS Information Systems Div.
801 National Center, Reston VA 22092
Dr. J.T. Young, Space Sciences Engineering Center
University of Wisconsin, Madison

We have also tried to stay in contact with a number of individuals at the various NASA facilities, including:

Ames Research Center:
Mr. James Brass, Dr. James Lawless - Ecosystems Branch
Mr. Gary Shelton - Aircraft Program
NASA Headquarters:
Dr. James Dodge
Dr. Eni Njoku
Dr. Paul Smith
Mr. Tony Villasenor
Mr. Jim Weiss
Mr. Phil Zion
Jet Propulsion Laboratory:
Dr. Rom Blom
Dr. Tom Logan
Dr. Chuck Klose (NODS)
Dr. Victor Zlotnicky (NODS)
Goddard Space Flight Center:
Mr. Jim Green (NSSDC, Data Systems Lexicon)
Dr. Forest Hall (FIFE)

Browse in the EOS Era

Marshall Space Flight Center
Mr. Michael Goodman (WETNET)
Dr. Greg Wilson (WETNET)

In terms of longer-term research and development during the second year, we emphasized several work areas. We have ported the majority of the BROWSE testbed to an IBM PC/AT, using Oracle as the embedded DBMS. Graphics were created by porting the MAPWD Fortran source to Borland's Turbo-C. The following pages present the database schema used in this port, as well as sample relations.

BROWSE SCHEMA – ORACLE VERSION

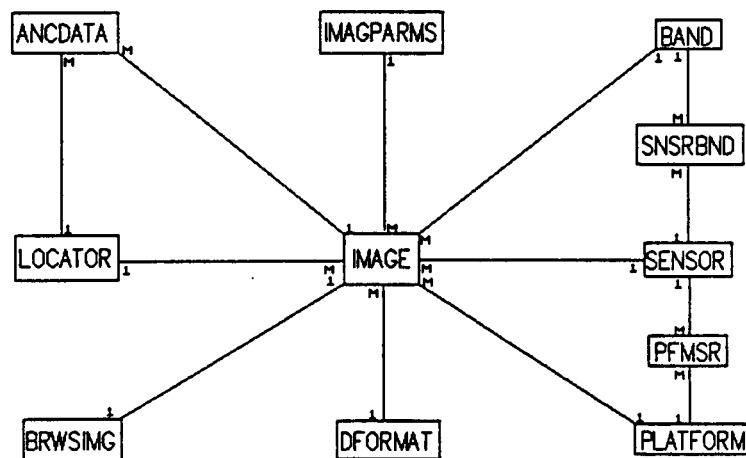


IMAGE	BAND	SENSOR	PLATFORM	LOCATOR	ANCDATA
SCENED	BANDID	SENSORID	PLTFRMID	DATAID	DATAID
DATEAQ	SPECRES	NAME	NAME	GEODESC	GEODESC
CLOUDCOV	RADRES	SENSRTYP	MISSION	LATDEG	DATATYPE
GEODESC	SPATRES	ORIENTATION	SDATE	LATMIN	SCALE
DATAQUAL	FREQ	NUMBDS	EDATE	LONGDEG	SOURCE
ONLIN	POLARZTN	SWATH	ALTITUDE	LONGMIN	LOCATION
IMAGPARM	DEPANGLE	DESCR	ORBIT		REFERENCE
FORMT	SCHANNEL		REF		
SENSOR			AGENCY		
BAND			OVERPASS		
PLATFORM					
PATH					
RW					
ADDRESS					

IMGPARMS	SNSRBND	PFMSR	DFORMAT	BRWSIMG
PARMID	SENSOR	SENSOR	FROMATID	SCENED
NUMLINS	BAND	PLATFORM	FORM	PROCESSING
NUMSAMP			MEDIUM	ADDRESS

PLATFORM:

PLTFRMID	NAME	MISSI
1	NOAA	6
2	NOAA	7
3	DMSP	5D-2
4	SEASAT	1
5	LANDSAT	1
6	LANDSAT	2
7	LANDSAT	3
8	LANDSAT	4
9	LANDSAT	5
10	SPOT	1
11	HCMM	A

PLTFRMID	NAME	MISSI
12	SPACE SHUTTLE	
13	AIRCRAFT	
14	DMSP	5D-2
15	NIMBUS	7

15 records selected.

SENSOR:

SENSORID	NAME
1	AVHRR
3	COASTAL ZONE COLOR SCANNR
4	RETURN BEAM VIDICON
5	MULTISPECTRAL SCANNER
6	THEMATIC MAPPER
7	HIGH RESOLUTION VISIBLE
8	HEAT CAPACITY MAPPING MIS
9	SEASAT SAR
10	SHUTTLE IMAGING RADAR
11	SHUTTLE-IMAGING RADAR-B
12	THEMATIC MAPPER SIMULATOR

11 records selected.

40 SIR-B/SAR
 41 SIR-B/SAR
 42 TMS/1
 43 TMS/2
 44 TMS/3

BANDID SCHANNEL

 45 TMS/4
 46 TMS/5
 47 TMS/6
 48 TMS/7
 49 TMS/8
 50 TMS/9
 51 TMS/10
 52 TMS/11
 53 TMS/12

53 records selected.

PLATFORM-SENSORS:

PLATFORM	SENSOR
-----	-----
1	1
2	2
15	3
5	4
6	4
7	4
5	5
6	5
7	5
8	5
9	5

PLATFORM	SENSOR
-----	-----
8	6
9	6
10	7
11	8
4	9
12	10
12	11
13	12

19 records selected.

SENSOR-BANDS:

SENSOR	BAND
1	1
1	3
1	4
1	2
1	5
1	6
3	7
3	8
3	9
3	10
3	11

SENSOR	BAND
3	12
4	13
4	14
4	15
4	16
5	17
5	18
5	19
5	20
5	21
5	22

SENSOR	BAND
5	23
5	24
5	25
6	26
6	27
6	28
6	29
6	30
6	31
6	32
7	33

SENSOR	BAND
7	34
7	35
7	36
8	37
8	38

9	39
10	40
11	41

41 records selected.

Browse in the EOS Era

As mentioned in previous reports, we prepared documentation for both the testbed itself, as well as the PC-based image display software created during this project. Copies of both of these documents appear in the appendices of this report.

Through our collaborations with the Research Libraries Group, we are participating in an external system design for a comparable system that will handle map and other non-remotely sensed data in addition to the image data now our focus. After this is completed, we will examine the problems of distributed heterogeneous data base query. We are also consider simple expert system approaches to reduce the level of expertise a user must have to be able to gain value from the BROWSE testbed.

In conclusion, this report documents the work we have accomplished during the first contract year of NASA Grant NAGW-987, **Browse in the EOS Era**. We include copies of papers written and presentations made during the year, a status report on the existing testbed, and an indication of the directions we wish to pursue in the next contract period. We look forward to the developments to come, and our continuing collaboration with our science advisory team.

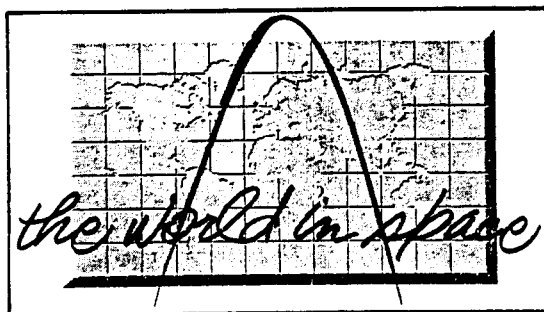
III. Papers and Presentations

Knowledge-Based Image Data Management: An Expert Front-End for the Browse Facility

Browse Capability for Spatial Databases

Electronic Browsing for Suitable GIS Data

MAPWD: An Interactive Mapping Tool for Accessing Geo-Referenced Data Sets



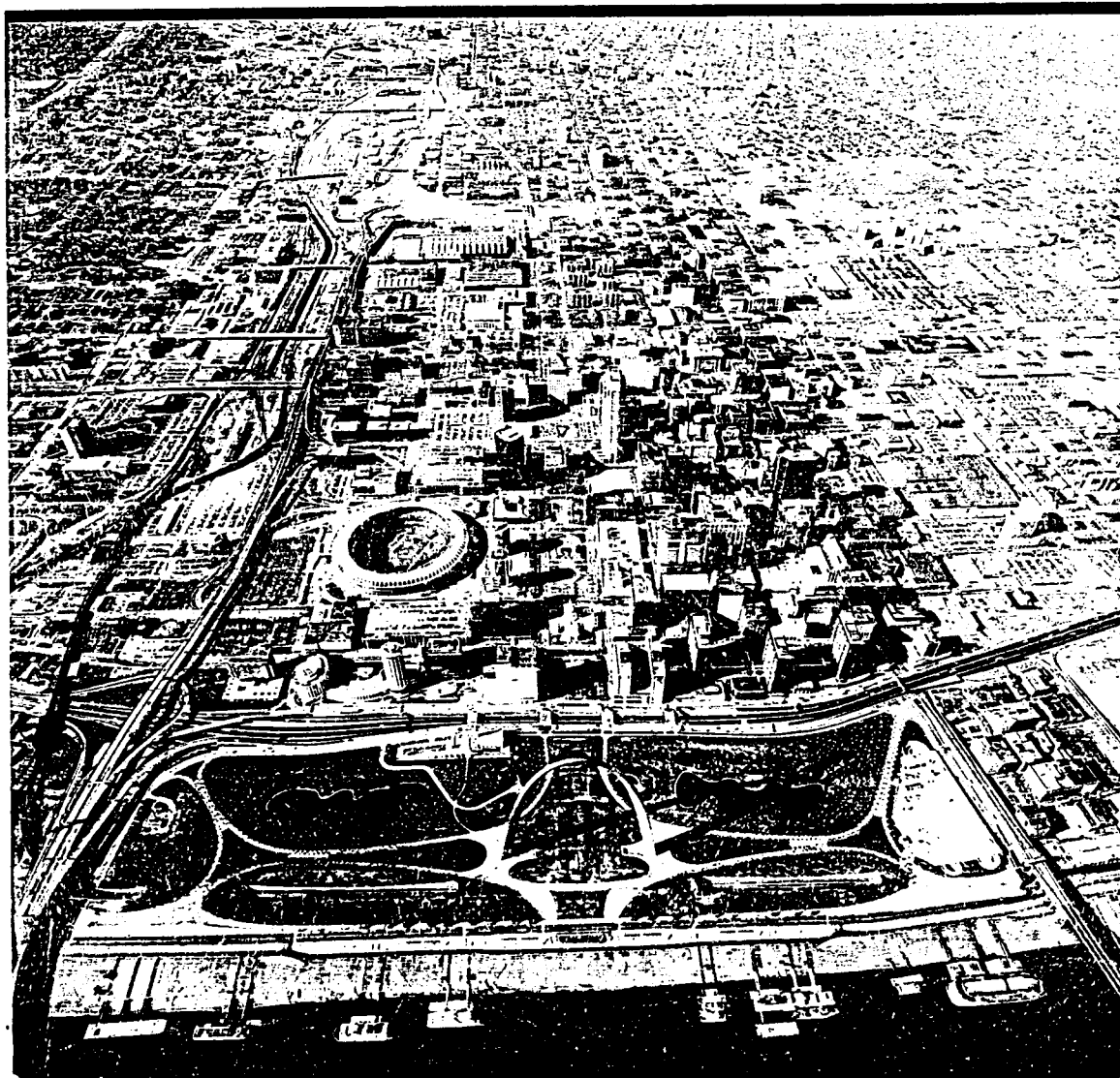
ACSM • ASPRS/1988 THE WORLD IN SPACE
MARCH 13-18/ST. LOUIS

David Storms
**TECHNICAL
PAPERS**
1988 ACSM-ASPRS
ANNUAL CONVENTION

VOLUME 4



IMAGE PROCESSING/ REMOTE SENSING



ORIGINAL PAGE IS
OF POOR QUALITY

KNOWLEDGE-BASED IMAGE DATA MANAGEMENT: AN EXPERT FRONT-END FOR THE BROWSE FACILITY

David M. Stoms, Jeffrey L. Star, and John E. Estes

*Remote Sensing Research Unit
Department of Geography
University of California
Santa Barbara, CA 93108
(805)961-3845*

BIOGRAPHICAL SKETCHES

David M. Stoms is a Ph.D. student in the Department of Geography at the University of California, Santa Barbara. Prior to graduate school, he worked 12 years with the U.S. Forest Service, primarily in land management planning. His current research interests are in browsing capabilities for spatial databases, knowledge-based approaches to spatial data processing and retrieval, and the value of improved spatial information.

Dr. Jeffrey L. Star is manager of the Remote Sensing Research Unit and Lecturer at the University of California, Santa Barbara. He received his undergraduate degree from M.I.T. and his doctorate in oceanography from Scripps Institution of Oceanography, University of California, San Diego. His research interests are in remote access to large archives of spatial data, merging artificial intelligence with geoprocessing technologies, and creation of a regional GIS for agricultural monitoring. Dr. Star has served on a number of national committees, including NASA's Earth Observing System Data Panel and the Task Force on Geo Information of the Research Libraries Group.

Dr. John E. Estes is Professor of Geography and Director of the Remote Sensing Research Unit at the University of California, Santa Barbara. He has authored more than 250 works on remote sensing and information systems, including 'Fundamentals of Image Analysis', a chapter in the *Manual of Remote Sensing*, which he coedited. He has served as senior scientist at the National Aeronautics and Space Administration and is a member of the Committee on Data Management and Computation of the National Academy of Science.

ABSTRACT

An intelligent user interface is being added to the NASA-sponsored BROWSE testbed facility, developed at the University of California, Santa Barbara, Remote Sensing Research Unit. BROWSE is a prototype system to explore issues involved in locating image data in distributed archives and displaying low resolution versions of that imagery at a local terminal. The original user interface for BROWSE incorporated knowledge of the data base structure and query language. Users were questioned on their choice of sensors and other attributes, and then BROWSE would formulate the appropriate query. The new feature described in this paper provides expert consultation to researchers not trained in remote sensing concepts. Users are asked about their objectives, and the interface matches the objectives with the database domains of image attributes. Natural vegetation applications are the focus of the initial development. Expertise in remote sensing of forest and range land is being extracted from the literature for use in the knowledge-base. This paper describes work in progress on development of the expert system. Sharing of expertise should encourage a broader research community to utilize remotely sensed data.

INTRODUCTION

Launch of the Earth Observing System (Eos) sensor packages as part of the Space Station complex in the 1990's will magnify many of the image data management issues. Eos has been projected to generate an unprecedented terabyte of image data per day for at least a decade. Global science questions can then be meaningfully addressed by researchers in many disciplines, including some who have not been involved with remotely sensed data before. Even for experienced remote sensing scientists, finding the right dataset can be a significant investment of time. Furthermore, the data must often be ordered on the basis of a terse, sometimes unreliable, database record of their quality. One approach to these shortcomings has been to promote a browsing capability in the data systems (NASA, 1986; CODMAC, 1982; CODMAC, 1986; Billingsley, 1984).

Browsing is defined as the process of locating and viewing image data from a remote terminal (Star et al, 1987; Star et al, in press). A prototype browsing system for image data has been developed at the University of California, Santa Barbara (UCSB) under a grant from the National Aeronautics and Space Administration (NASA). The BROWSE testbed basically integrates a database management system (DBMS), a user interface, communications software, and an image display system. The user is expected to select the appropriate attributes and their values.

The Eos Data Panel, as well as other authors, have recommended the use of expert system technology to improve database access, particularly by new and infrequent users (NASA, 1986; Estes et al, 1986). An intelligent user interface, or front-end, is a program that will 'serve as an intermediary between a database and a user who has little or no knowledge of the database's architecture, language, or content' (Campbell et al, in press). NASA anticipates several modes of browsing for Eos data, requiring different types of interfaces. The first mode, handled by the initial version of BROWSE, visually confirms the quality and coverage of the image

found in the Catalog. Other users will browse for spatial objects extracted from highly processed data. The third form of browsing will be by researchers who may not be experts in remote sensing but who need inputs derived from digital imagery. We are attempting to support this third group with a new intelligent interface. Specifically, we want to evaluate two objectives: 1) can expertise in remote sensing capabilities be formalized in a knowledge-base, and 2) would such a sharing of expertise be useful to non-experts in selecting image and spatial data? In other words, whereas the original BROWSE system incorporated only a low level knowledge of the data base structure and contents, the new interface adds a higher level of knowledge of the spatial, spectral, and temporal parameters suitable for a variety of tasks in a relatively limited domain.

Following a brief overview of the BROWSE testbed, the related concepts of browsing and intelligent interfaces in information science are reviewed. This explanation sets the foundation for discussing the intelligent user interface being developed for the BROWSE testbed. For prototyping, the initial application is remote sensing of forest and range land, where a large knowledge-base has accrued in the research literature. There are many potential parameters of vegetation that can be derived from image data, and a large number of image sources have been used in the past (Carnegie et al, 1983; Heller, 1985). Furthermore, many of the global science issues such as deforestation, desertification, climatic change, carbon cycling, and acid rain involve vegetative factors (Billingsley, 1984; NASA, 1984).

THE BROWSE TESTBED

The prototype BROWSE system was developed under the auspices of a NASA grant to explore formats most suitable for viewing spatial data by a range of multidisciplinary scientists. However, we have found the display of browse images is irrevocably intertwined with the process of retrieving appropriate data.

The hub of the BROWSE testbed is a relational data base management system. The CATALOG data base describes the attributes, or metadata, of each discrete image, such as its processing history, image quality, and so on. In the original prototype, a user interface was provided to guide a researcher in the query process. We took this approach to save the user from needing to learn the DBMS language and the structure of the data base. An investigator is presented a series of menus and prompts for selecting the query attributes. If a user finds an image that appears to meet their needs, based on a detailed record, they can transmit a browse file from the host computer to their local terminal for display. A fuller description of the BROWSE testbed, as well as some of the potential browse formats, are given in Star et al (1987 and in press).

After our experience with the prototype system and the reaction of our scientific collaborators, we feel that the display process can not be divorced from the query process when evaluating the effectiveness of a browse facility. For instance, our use of a multiresolution electronic map to interactively identify a geographic area of interest makes the overall browse process easier. (See Figure 1 for an example of this graphic feature). For users who are relatively new to remote sensing, i.e., the wider constituency that NASA hopes to attract, an effective integration of locating and displaying data will be even more important. The original testbed

provides only a low level of intelligence, or cleverness, in helping a user bridge the gap from what they want to what is in the data base. That is, the knowledge-base only concerned the DBMS language and the data base contents. The user had to supply their own knowledge of the appropriate type of sensor, best dates of coverage, and so on. The new interface provides an additional level of remote sensing expertise to translate user objectives into the data base attribute domain.

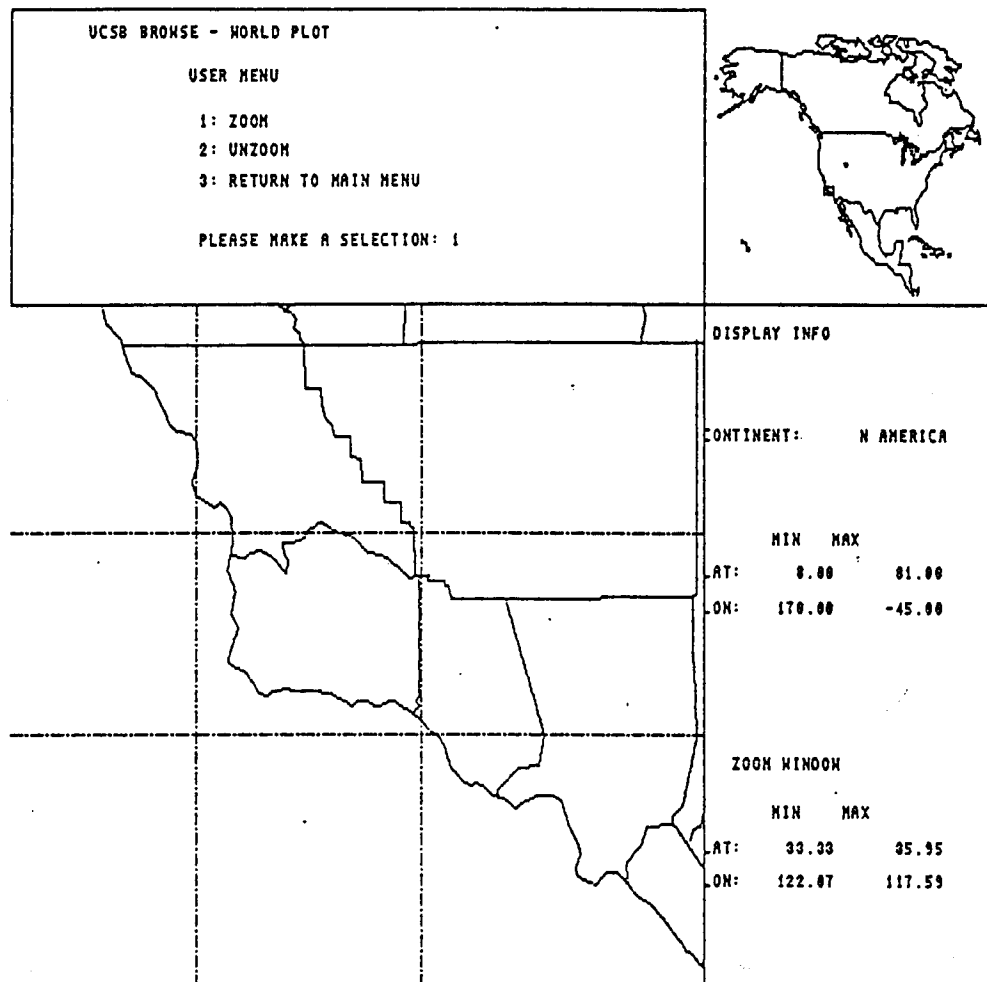


Figure 1. Example of Electronic Map for Selecting Geographic Search Area.

BROWSING FROM THE DATA MANAGEMENT FIELD

In the data base management field, *browsing* refers to the process of accessing information when the user is unfamiliar with the data base contents and its organization. Furthermore, the user is not required to predefine a precise data object as the goal of their search (Godin et al, 1986; Robertson et al, 1985; Monarch et al, 1987; Estes, 1986). Consequently, a browser is a free-form means of interacting with a database. A DBMS with a browse feature assists the user in gradually narrowing the focus of the search, suggesting attributes, related objects, asking appropriate questions, just as a human expert might. In other words, the browsing mechanism has knowledge of the application domain as well as of the data base structure. This section summarizes a few browsers in database systems similar to our testbed. A key component of these examples is the concept of *views*. Users

have one or more views of a database that stems from their specialty, which determines what they expect to see. Simultaneously, there is an architectural view of how the database is structured. Thus, browse facilities and intelligent interfaces are closely related techniques.

ECO Browser is an intelligent front-end designed to manage and retrieve a loosely structured body of ecological observations for ecologists designing dynamic models (Robertson et al, 1985). Because ecology is such a diverse field (one might say that there are many potential user views of the data base), the browse mechanism was kept independent of any specific ecological context. The user is presented a graphic display of position in the hierarchical search space. In the initial version, the user was given no advice from the interface about which attributes were significant to the given objective.

The RABBIT intelligent database interface (Tou et al, 1982) uses what the authors called *retrieval by reformulation*. RABBIT displays part of a single record, showing only the values for the attributes selected so far. The user then *reformulates* the query by modifying or adding to the search criteria based on this partial record. As with ECO Browser, the user must decide whether the data is suitable.

The Intelligent User Interface (IUI) is being developed at the NASA Goddard Space Flight Center. (Campbell et al, in press). While not specifically called a browser, the IUI is designed to support users with little knowledge of the contents, structure, or query language of NASA's scientific databases. Conceptual graphs define a path, employing an expert system, from the user view based on their domain knowledge, onto the architectural view of how the data base is organized.

REMOTE SENSING OF FOREST AND RANGE LAND

Analyzing vegetation cover with airborne sensors has a history of over a 100 years. In the 1880's, foresters used cameras mounted on captive balloons to map forests in Germany. Over the last century, a substantial body of experience has accumulated. Many aspects of vegetation cover have been measured or derived by remote sensing techniques. Categorizing these parameters into mapping, monitoring, and modeling (Estes, 1986), we can include those shown in Table 1.

Vegetation is a complex resource, or one could say, there are multiple views of vegetation. Forest and range land can be viewed as a commercial resource for timber and grazing, as habitat, fuel, watershed, recreation, esthetics, or as a link in nutrient cycles, not to mention its influence on climate. On the other hand, vegetation has been observed by a wide variety of sensors, both passive and active, throughout the electromagnetic spectrum, at many spatial and temporal resolutions. Few individuals can master all the relevant disciplines. What we face then is a type of application that Feigenbaum and McCorduck (1983) describe as most suitable for expert system development. The expertise is distributed among a relatively small number of individuals and needs to be fused in a knowledge-base to become more comprehensive, up-to-date, and accessible to non-experts.

Table 1. Vegetation Parameters Derived from Remote Sensing

Mapping	Stand Type Stand Structure Density or Cover Tree Height Fuels Wildlife Habitat
Monitoring	Change Detection Range Trend and Condition Deforestation/Desertification Stress Insect/Disease Attacks Fire Detection
Modeling	Biomass/Yield/Volume Leaf Area Index Productivity Potential Vegetation Evapotranspiration

THE BROWSE KNOWLEDGE-BASE

With such a massive body of knowledge, in terms of both the vegetation and the remote sensing parameters, we had to select a manageable subset of the field. We initially picked applications relevant to the vegetation of California, being an area with a very diverse mix of landscapes, and one the authors are most familiar with. Papers published in the professional journals and conference proceedings were summarized into database records. Each article was abstracted into vegetation and image attributes. For example, the data extracted from three of these papers are displayed in Table 2. The vegetation based factors correspond to a user view of a database, whereas the image attributes correspond more to an architectural view of the BROWSE catalog. A factor for ecosystem type was included to match research results from similar environments when it is not available explicitly for a given geographical region.

The next step was to convert this database of case studies into a knowledge-base of expert system rules. We have initially chosen to use an expert system shell, a software tool to develop knowledge-bases, rather than program our own expert system from scratch. The reason for this was that our objective was to attempt to represent the knowledge in an expert system format and not to worry about inference and representation details at this time.

One of the tools we are trying is VP-Expert, a commercial PC-based expert system shell that uses backward chaining inference to process its rule base (Paperback Software, no date). Backward chaining establishes a search goal, and then backtracks through the rules until facts are found that support the conclusion. The goal,

Table 2. Example Attributes Extracted from Three Research Papers			
Author	Franklin	Yool et al	Logan
Vegetation Factor	1. Type map 2. Density 3. Biomass	Fuel type	Biomass
Location	No. California	So. California	No. California
Ecosystem Type	Temperate Conifer Forest	Sclerophyllous Chaparral	Temperate Conifer Forest
Min.Mapping Unit	not given	100 acres	not given
Scale	Regional	Regional	Ecosystem
Platform	Aircraft	Landsat	NOAA
Sensor	TMS	MSS	AVHRR
Date	May 84 (incidental)	June 78 (minimize shadows)	Sept 80 (avoid snow)
Time of Day	09:30	09:30	15:00
Best Spectral Bands	1. NIR, Thermal 2. Red 3. Red	Red, NIR	Red
Role of Data	1. Type class 2. Reflectance 3. Reflectance	Type class	Reflectance

(Franklin, 1986; Yool et al, 1985; Logan, 1983).

or set of goals, in our case are the appropriate data attributes of platform, sensor, time of year and of day, and spectral coverage. A non-expert is not expected to be able to answer questions about the choice of attributes directly. Instead, the expert system asks about the objectives of the study such as the vegetation factors to be derived, in what ecosystem type, and at what minimum mapping unit size. The user's answers supply the facts needed to trigger the appropriate rule.

A positive feature available in VP-Expert is to allow users to ask why a particular question is being asked of them. The default response to a *why* question is to display the rule in the knowledge-base that triggered it. A superior approach we will use is to provide a more technical answer. For example, if the original question asked by the expert system is *'What is your minimum mapping unit size?'*, the *why* response might be *'To detect objects, the spatial resolution of the sensor must be no more than half of the minimum mapping unit size'*. We believe this

approach serves as a tutorial so that new users can become more knowledgeable.

VP-Expert has a facility called INDUCE that automatically generates an initial knowledge-base from a table of attributes. The records of the literature summaries, as seen in Table 2, were put into a table. VP-Expert induces the rules from the table. Then we edit the results, adding answers to *why* questions, polishing the wording of the prompts, and enhancing and verifying the rules. Multiple answers are allowed, if, for instance, two or more sensors can both achieve the same task. Our approach will be to use half of the literature sources to generate the rules and the other half to test them. One question we will be asking is how many examples are needed to develop comprehensive rules. Induction is the technique often used in machine learning experiments (Michalski et al, 1980).

The initial prototype is being designed as a stand-alone system on the PC. The user only gets the appropriate image attributes and may then log on to the BROWSE facility, asking for data with those attributes. Ideally, the expert system will be integrated into BROWSE. Then the system could directly pass the selected attributes to the DBMS. A further issue to be addressed at that time will be how to respond when no data is found. If no imagery is retrieved from the database, the system needs to be intelligent enough to 1) suggest alternative archives that might have the data, or 2) suggest queries based on similar attributes (Billingsley, 1984). Defining similarities of ecosystem type, spectral and spatial resolution, geographic location, and so on, will be a major challenge since it may turn out to be very much application specific. However, a human expert would be expected to cope with just this type of ambiguity.

CONCLUSIONS

Research experience turns out to be difficult to formalize. It is difficult to be comprehensive in what results to include and still be selective for genuine expertise. Thorough testing by real experts will be necessary to refine the rules.

The initial prototype under development follows a rather rigid structure. Users are given a choice of responses to each question that correspond to the conditions in the rules. An '*I don't know*' choice is provided when appropriate, assuming that a user may not have a clearly defined task. However, the terms used for the choices may not match exactly what the user wants to do, or the terms themselves may be imprecise. A faculty to interpret natural language textual input of the user's objectives would be a friendlier mode of interacting. However, natural language processing is a complex issue (Robinson et al, 1985) and beyond the scope of the current project.

The intelligent user interface outlined above serves only one of several potential classes of browsers. Users that will not benefit from our initial prototype include those looking for specific image contents (e.g., mountain bark beetle infestations in lodgepole pine forest). In that case, highly processed data (level 2 or higher) is required and the object types stored as image attributes (NASA, 1986). In that situation, searching would be done on spatial objects rather than on image characteristics. Complex, or virtual objects, not stored directly could be defined from simpler objects (Campbell et al, in press). However, the complexity of extracting

multidisciplinary objects from imagery acquired at the rates planned for Eos would be a staggering task. While an exciting prospect, browsing on image contents must be approached cautiously.

Although we are still in the early stages of development of an intelligent user interface for the BROWSE testbed, we are encouraged that capturing remote sensing knowledge as an expert system is achievable and appropriate. The knowledge currently is distributed among a relatively small number of experts. Fusing knowledge from many experts and disseminating it as an expert system could encourage additional scientists to utilize remotely sensed data in their applications. Scientists can then focus more on their research and less on becoming database management and remote sensing experts (Estes, 1985).

ACKNOWLEDGEMENTS

Research for this project is being sponsored under NASA Grant NAGW-987. The authors would also like to thank Dr. Alan Strahler for a stimulating discussion on knowledge-based interfaces.

REFERENCES

- Billingsley, Fred C., 1984, "Data System Considerations for Remote Sensing," in *Proceedings of SPIE*, vol. 475, pp. 117-128.
- Campbell, William J. and Larry H. Roelofs, in press, "The Development of an Intelligent User Interface for NASA's Scientific Databases," *Proceedings of the American Association for the Advancement of Science Meeting*, Philadelphia, PA.
- Carnegie, David M., Barry J. Schrumph, and David A. Mouat, 1983, "Rangeland Applications," in *Manual of Remote Sensing*, ed. Robert N. Colwell, vol. II, pp. 2325-2384.
- CODMAC, 1982, *Data Management and Computation*, 1: Issues and Recommendations, p. 167, National Academy Press, Washington, D.C..
- CODMAC, 1986, *Issues and Recommendations Associated with Distributed Computation and Data Management Systems for the Space Sciences*, p. 111, National Academy Press, Washington, D.C..
- Estes, John E., 1985, "The Need for Improved Information Systems," *Canadian Journal of Remote Sensing*, vol. 11, pp. 124-131.
- Estes, John E., 1986, "A Perspective on the Use of Geographic Information Systems for Environmental Protection," *Proceedings of a Conference on Geographic Information Systems for Environmental Protection*, pp. 3-23, Environmental Protection Agency, Las Vegas, NV.
- Estes, John E., Charlene Sailer, and Larry R. Tinney, 1986, "Application of Artificial Intelligence Techniques to Remote Sensing," *The Professional Geographer*, vol. 38, pp. 133-141.
- Feigenbaum, Edward A. and Pamela McCorduck, 1983, *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*, p. 275, Addison-Wesley, Reading, MA.

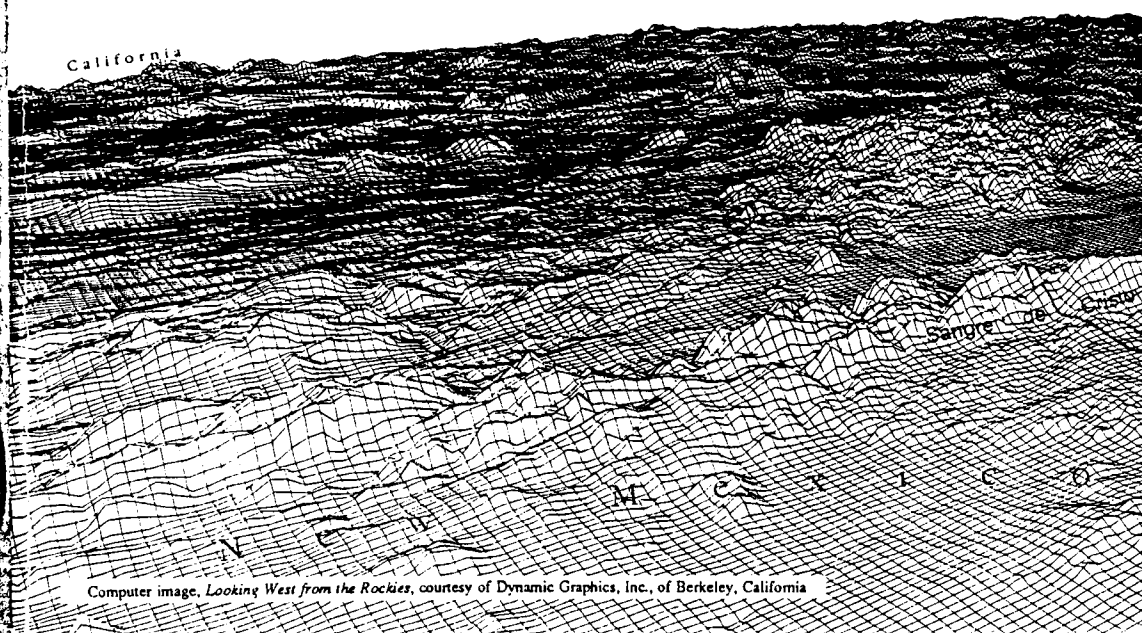
- Franklin, Janet, 1986, "Thematic Mapper Analysis of Conifer Forest Structure and Composition," *International Journal of Remote Sensing*, vol. 7, pp. 1287-1301.
- Godin, Robert, Eugene Saunders, and Jan Gecsei, 1986, "Lattice Model of Browseable Data Spaces," *Information Sciences*, vol. 40, pp. 89-116.
- Heller, Robert C., 1985, "Remote Sensing: Its State-of-the-Art in Forestry," in *Proceedings of Pecora 10: Remote Sensing in forest and Range Resource Management*, pp. 18-29, Ft. Collins, CO.
- Logan, Thomas L. 1983, , *Regional Biomass Estimation of a Coniferous Forest Environment from NOAA-AVHRR Satellite Imagery*, p. 263, University of California, Santa Barbara. PhD Dissertation
- Michalski, R. S. and R. L. Chilausky, 1980, "Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis," *International Journal of Policy Analysis and Information Systems*, vol. 4, pp. 125-161.
- Monarch, Ira and Jaime Carbonell, 1987, "CoalSORT: A Knowledge-Based Interface," *IEEE Expert*, vol. 2, pp. 39-52.
- NASA, 1984, *Earth Observing System--Science and Mission Requirements Working Group Report*, I, p. 58.
- NASA, 1986, *Earth Observing System--Data and Information System: Report of the Eos Data Panel*, IIa, p. 62.
- Paperback Software, no date, *VP-Expert: Rule-Based Expert System Development Tool*.
- Robertson, David, Robert Muetzelfeldt, Dave Plummer, Mike Uschold, and Alan Bundy, 1985, "The ECO Browser," *Proceedings of the Fifth Technical Conference of the British Computer Society--Specialist Group on Expert Systems*, pp. 143-156, Cambridge University Press.
- Robinson, Vincent B., Dominique Thongs, and Matthew Blaze, 1985, "Natural Language in Geographic Data Processing Systems," in *Proceedings of the International Conference on Advanced Technology for Monitoring and Processing Global Environmental Data*, pp. 67-73.
- Star, Jeffrey L., Mark A. Friedl, David M. Stoms, and John E. Estes, 1987, "Browse Capability for Spatial Databases," *Proceedings of GIS'87*, pp. 196-205, San Francisco, CA.
- Star, Jeffrey L., David M. Stoms, Mark A. Friedl, and John E. Estes, in press, "Electronic Browsing for Suitable GIS Data," *Proceedings of IGIS Symposium*, Crystal City, VA.
- Tou, Frederick N., Michael D. Williams, Richard Fikes, Austin Henderson, and Thomas Malone, 1982, "RABBIT: An Intelligent Database Assistant," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 314-319, Pittsburgh, PA.
- Yool, Stephen R., David W. Eckhardt, John E. Estes, and Michael J. Cosentino, 1985, "Describing the Brushfire Hazard in Southern California," *Annals of the Association of American Geographers*, vol. 75, pp. 417-430.



GIS '87 – SAN FRANCISCO

Second Annual International Conference,
Exhibits and Workshops on Geographic
Information Systems

ORIGINAL PAGE IS
OF POOR QUALITY



BROWSE CAPABILITY FOR SPATIAL DATABASES

Jeffrey L. Star, Mark A. Friedl, David M. Stoms,
and John E. Estes

Geography Remote Sensing Unit
Department of Geography
University of California, Santa Barbara
California, 93106

ABSTRACT

A need exists to develop superior data management and information systems for remote sensing and other spatial data sets. This issue will become even more significant in the near future as remote sensing instrumentation continues to produce ever increasing volumes of data, and scientists attempt global scale earth science investigations. A testbed information system has been created to implement and evaluate some proposed solutions to future data management issues. In particular, strategies to maximize user access to a wide variety of data sources, and large scale spatial database design are being considered. By implementing a prototype system valuable insight will be gained for spatial information system design in the future.

INTRODUCTION

Earth science and related technology have progressed to the point where the conduct of global science now appears feasible (Estes and Star 1986). This is reflected by the growing interest of scientists from a wide range of disciplines in performing global scale investigations related to the dynamic coupling of the lithosphere, hydrosphere, biosphere, and atmosphere. A central element of such global scale studies is the utilization of advanced, high resolution remotely sensed data. An important component in this regard is the suite of sensors included in the Earth Observing System (EOS) planned for the space station complex of the 1990's (Arvidson et al 1985). This system will provide immense volumes (in the order of up to one terabyte of data per day) of high quality data to the scientific community.

A critical issue which needs to be addressed in this area is the effective management of data derived from EOS era sensor systems. Included in this domain are questions related to both technical considerations (ie. hardware and software) as well as strategic considerations dealing with the management of the data. Also, methods of incorporating image data and other non-image spatial data into a single database environment will be required for geo-based data management and analysis systems. Currently, much sophisticated database and information management

technology is available for such tasks. However, superior architectures and well considered data management strategies are necessary if the data volumes derived from EOS era sensor systems are to be effectively used for earth system science applications.

The research described in this paper represents a preliminary attempt to clarify and resolve some of the issues identified above. Rapid prototyping approaches have been employed to create a testbed system (entitled BROWSE) in which user needs and data management strategies for large scale scientific spatial databases can be evaluated. The testbed consists of a data base of spatial (primarily image) data, and associated software which will allow the user to select data sets of interest, and to preview image data on a microcomputer workstation. The emphasis of this testbed is geared towards resolving data management issues in contrast to more technical considerations related to complex software implementations.

TESTBED ENVIRONMENT

Objectives and Goals

In 1982, the Committee on Data Management and Computation (CODMAC 1982) recommended that:

"NASA evaluate and develop the concept of 'electronic browse' capability, allowing users to explore data files via communication links. We recommend that 'quick-look' low resolution data be included for use in electronic data browsing."

Using this type of data access technology, earth scientists could potentially have much faster and easier access to data than is currently possible. The BROWSE project is examining this type of technology and is attempting to formalize the definition of so called "electronic browse capability."

There are currently several very large data centers across North America which are being used to archive remotely sensed data from both airborne and satellite mounted sensors (eg. EOSAT, NASA Ocean Data System, Canada Center for Remote Sensing, EROS Data Center, etc). This situation has several problems which are either presently an issue or will be so in the near future. Despite a wealth of data and information available at each site, there exists very little communication and exchange of resources between these centers. To a large degree, this situation has arisen due to the use of incompatible systems (both hardware and software) at each site. However, current networking and database technology has significant potential to overcome much of the bottleneck in this area.

The BROWSE project is examining implementation strategies which maximize user accessibility to remote sensing and other ancillary data sets. The research is specifically being conducted with a multidisciplinary orientation. In this regard, the system has been designed with several key

objectives in mind. Users must be able to access the database from remote locations using user friendly software and widely available hardware. Having accessed the database a user should then be able to locate appropriate data sets, and preview subsetting images at his or her workstation. Implicit in these objectives are the requirements to define a basic set of attributes which can be used to query the database, and also the identification of a concise set of processed (eg. vegetation greenness indices, principle components, etc.) and subsetting imagery which can be employed by the user to select useful data sets for a specific application.

A second major issue is the management of large data volumes produced by remote sensing platforms. Current sensor systems have already outstripped the ability of data management technology to cope with the large volumes of data which are being produced. Clearly, a need exists to develop data management technology and networking strategies which are able to handle much larger data volumes, and which facilitate access to this data by as wide an audience as possible.

In this context, there is presently much interest in both developing strategies for archiving data sets, as well as investigating networking technology to maximize user access to remote sensing data sources and reduce data redundancy between repositories. There has been a growing awareness among the remote sensing community that the archiving of data will be an important area requiring careful consideration in the near future. Data management strategies will be central to this discussion. In this regard, decision strategies related to the deletion and retention of data sets in geographic databases will be fundamental.

Concurrently, there is a distinct trend towards the creation of discipline specific information systems (ie. land and climate, atmospheric, oceanographic, etc.) for the purpose of data redundancy reduction and modularity of information content. Networking technology to maximize communications between these various systems will be essential if the data is to be utilized in an efficient manner. The ultimate goal of such a network system is to provide a distributed database of remotely sensed and other ancillary data, providing the applications scientist with maximum access to a wide variety of data sources.

The BROWSE project is emphasizing specific components within this framework. In particular, database design for large scale image databases is being examined and optimal strategies for developing distributed spatial databases are being considered. The key objective in this regard is to consider requirements for information systems which maximize user accessibility to data sources. In this context, a prototype image database has been created which will serve as the model for one node within a distributed image database network. Using this test site, design strategies for large scale image databases are being

implemented and evaluated. After the prototype image database has been populated, communications software to link the BROWSE system with other remote sensing information systems will be developed. In this manner, the concepts of spatial database management may be tested.

Database Management Technology

Database management and information system technology has been a field of active research for the past twenty years. Current database management systems (DBMS) and their associated data models are highly sophisticated and have proven to be extremely useful tools for many purposes. Unfortunately, these systems have been developed primarily for commercial applications. Consequently, they are not designed for handling spatial (or geo-referenced) data. A primary goal of BROWSE is to develop methodologies which effectively adapt existing DBMS technology to handle spatial data.

Several data models (hierarchical, network, relational) have evolved in database technology from a need to efficiently store and retrieve data and relationships between data. The evolution in this regard, particularly in terms of ad hoc queries, is the relational model. Unfortunately, in many cases the complex nature of spatial relationships make them very difficult to represent using conventional data models (Peuquet 1984). At the same time the tabular nature of the data representation employed in relational DBMS technology is ideal for storing and retrieving attribute data. Currently, there exists a high level of interest in investigating approaches to the relational model for spatial database applications (Lorie and Meier 1984, Palimaka et al 1986, Abel and Smith 1986).

BROWSE Design Methodology

The goals of BROWSE are to examine design approaches for browsing image databases, specifically for use by a multi-disciplinary group of users. Thus, the specific design attempts to incorporate as wide a range of data types as possible. In addition, the system is unrestricted in terms of geographic coverage, and is able to reference image data sets on a global basis. The specific user scenario for which this has been developed is that of the applications scientist accessing the database from a workstation in his or her office.

The approach which has been employed in BROWSE is to create a testbed environment which can provide valuable experience towards developing a data system which allows easy, integrated and complete access to past, present and future data sets. The primary component of the testbed is a database consisting of remotely sensed and other geo-referenced ancillary data sets. This database has been enhanced by a set of software utilities which aid the user in selecting appropriate data sets of interest, and to preview the image data once selected. In addition, a database of information regarding data centers and network nodes is included to assist the user in locating pertinent data. Together, this system provides a facility for

scientists from many disciplines at widely distributed locations to locate relevant data collections, browse available data sets, view image data, and identify existing ancillary data in order to facilitate a decision concerning acquisition of the data for the host site.

The final component of the system is a database which contains information about users, their specializations (both scientific and geographic) and system usage. A group of collaborators have been targeted within the remote sensing user community who will be acting as system users to provide feedback regarding the various implementation strategies which have been developed. In this manner, the testbed system can be monitored, evaluated and subsequently modified with respect to the specific data types and functions which have been included. This input will provide valuable insight regarding requirements for EOS era data management systems.

SYSTEM IMPLEMENTATION

Architecture - Hardware and Software

The BROWSE system architecture is divided into two logical components: hardware architecture and software architecture. As indicated above, the emphasis of the project is geared towards data management issues. However, to a large degree the overall success of the given strategies is dependent upon the system architecture. Therefore, technical issues must also be considered. The hardware architecture is composed of a host minicomputer (MicroVAX II), a remote microcomputer based work station (IBM PC-AT), and associated communications hardware and software. Using this configuration, the BROWSE system simulates the potential remote sensing application scientist who would use this system from his or her office. The specific hardware which has been selected for the project was chosen in order to be as compatible as possible with a diverse range of systems. Further, we have attempted to make the software transportable by relying on standard practices (eg. using standard Fortran 77) where possible. Thus, the end system will be accessible to a wide range of users.

The host minicomputer is acting as a dedicated database machine. Running under the VMS operating system environment, the system includes a database of remotely sensed and other spatial data, and descriptive information about that data. The database is managed using a public domain relational DBMS. Users are able to access the database from remote workstations using standard telephone line communications protocols. In this framework, the user is able to access descriptive as well as quantitative information regarding available imagery, and to quick view preprocessed images if desired.

In designing the system software for the BROWSE project, a rapid prototyping approach was employed. This methodology is a two stage process geared towards implementing a prototype system which will yield useful information for

designing a final system. The first stage of this process involves the development of a model which defines system requirements and design criteria. In the second stage the model is implemented and system performance is evaluated. The ultimate goal of this type of approach is not to produce a final system, but rather to implement a testbed environment which can be used to exercise ideas and to provide valuable experience and guidance towards implementing an on-line system with minimal design restrictions.

In designing the BROWSE system it was endeavored to use existing software wherever possible in order to minimize the burden of actual software development. A public domain relational DBMS (RIM - Relational Information Management) has been used to manage the spatial database. A menu driven front end to this DBMS has been written in Fortran in order to aid the user in specifying queries without having to learn the RIM command syntax. Both RIM and the user interface software are resident on the host mini-computer.

The second major software component of BROWSE is a set of utilities which are resident at the remote user workstation. These utilities include communications software, graphics software for defining geographic areas of interest, and image display software which allows the user to view subsetting images of selected data sets. Using these local workstation utilities in association with the database software resident at the host database node, the user is able to identify and preview potentially useful data sets for a specific application. Geographic areas of interest are specified interactively using graphical software to identify the area on a map plotted on the workstation screen. Available data sets for that area can then be selected based upon date, sensor, cloud cover, and other scene specific criteria. Finally, a selected image can be subsetting and previewed at the workstation using the image display software.

Database Design

The BROWSE database is simulating one node in a hypothetical distributed database of spatial data. Our primary focus is raster format remotely sensed data. Attribute information about the image data sets is included in the database, and queries are made based on these attributes. The database is composed of three major components: the DIRECTORY, the CATALOG, and the USER (table 1). The DIRECTORY consists of a top level database of information regarding the NODES in the distributed database, and a description of the holdings at each node. The CATALOG is a detailed database of the holdings at each NODE. The USER is a database of information regarding the various users of the system, their scientific and geographic specializations, as well as logging information which will allow the system designers to assess system usage and adjust to user needs. Thus, the final distributed database will consist of a set of NODES, each with unique CATALOG and USER databases representing one

entry in the DIRECTORY.

The DIRECTORY is resident at each node in the network and contains one entry for each node in the network. Each node is defined by a set of attributes which specify the types of holdings at that node. These attributes include the number of data objects available at the node, the geographic and scientific data emphasis of the node, and the principle collections which are present at the node and their format. By querying this database the user can identify that node or set of nodes which may contain relevant data sets for his or her specific application.

The CATALOG of each database is unique to the given node, and is a detailed inventory of the holdings of that node. Standard descriptive attributes about each image in the database can then be used to select specific images which are appropriate to the users needs. These attributes include: date and time of acquisition, platform, sensor, band, cloud cover, sun elevation, and any processing history which may have been performed on the data. In addition, the actual form of the data (ie. printed, film, or digital) at the node is included.

Top level	low level	joining categories
Directory	geographic coverage	principle collections
Catalog	scientific discipline	scientific focus (node)
User	sensor type	geographic emphasis
	band	sensor bands
	platform	platform sensors
	image parameters	discipline focus
	location	scientific focus (user)
	map reference	data emphasis
	data format	nodes queried
		sensors queried
		geographical areas

Table 1. table outlining the logical information content of the BROWSE database. In the actual database schema several of the low level and joining categories are shared by the Directory, Catalog, and User databases.

The USER database performs two main functions within the BROWSE system. First, this database contains information regarding remote sensing applications scientists, and their scientific and geographic specializations. Secondly, this database is used to perform system usage monitoring. In this way, users can obtain information regarding other scientists involved in related research. In addition, the system logging information can be used to evaluate system performance and design strategies for adjustment in future systems.

DATA ACCESS

The discussion to this point has emphasized implementation aspects of the BROWSE system. This is a reflection of the fact that to this date the majority of the research efforts have been devoted to system design and implementation. However, the ultimate emphasis of this project is to provide a remote sensing information system for applications scientists to aid in data set selection. The database and associated software are now in place and the system is currently in its initial phase of operation.

As previously indicated, a group of remote sensing applications scientists has been targeted to use and evaluate the system. Using the workstation software, these users may access the database using standard telephone line communication protocols. A toll free number has been established for this purpose. Using these facilities the user can query the database, down-load image data to his or her workstation, and display that data locally.

A typical session would first involve the initial dial up and login procedure. After accessing the BROWSE system, the user would then be prompted with a set of menus which can be used to specify attributes for a query of the database. The specific geographic area of interest is identified interactively by the user by graphically identifying a window on a map plotted on the workstation screen using a pointing device. The user is then prompted for more scene specific information. For example, a climatologist interested in land surface energy balance research in the Santa Barbara area might enter the following attributes:

Database of interest: CATALOG
platform: all
sensor: all
begin date: june 1, 1979
end date: sept 1, 1979

to which the system might respond with information of the following nature:

BROWSE database: 2 items found

1. Heat Capacity Mapping Mission,
July 3, 1979
3 bands online, 600m spatial resolution
scene id # A-A0062-16032-2
2. Landsat-3,
August 15, 1979
4 bands online, 79m spatial resolution
Scene id # 8246517502500

For more information please enter the scene id or ESC to escape:

If any of the items found are of interest to the user then

more detailed information (sensor specifications, resolution, number of lines and samples, sun angle, etc.) can be obtained by entering the specific scene id. In this way, the user can 'browse' the image database, and identify potentially useful data sets. If an appropriate image is found, the user can then exit the database and download the image data to his or her workstation. The image display software can then be used to view the actual image data. Therefore, the user has been provided with much valuable information regarding the acquisition of the actual data set for a specific application.

CONCLUSIONS

Superior data management and information system technology is required to efficiently manage the immense data volumes which are projected for EOS era remote sensing systems and other geo-referenced data sets in the 1990's. Included in this domain are issues related to spatial database management, distributed database and networking technology, data archiving strategies, and information systems technology which maximize user accessibility to data sources. The BROWSE system has employed rapid prototyping methodologies to implement and evaluate proposed solutions to some of the above problems. In particular, BROWSE has specifically examined questions pertinent to the access of spatial databases by users from remote locations. Issues which have been examined include database design, user accessibility and remote 'browsing' of image display on commonly available hardware.

To date, a prototype database has been designed and populated, and workstation software has been developed which allows a user to query the database and view image data from his or her office. The system is currently in its first phase of operation and is being used by a group of remote sensing applications scientists in order to evaluate system performance and features. After a preliminary evaluation period the system will be enhanced by providing linkages to other data centers and by providing an 'intelligent' user interface to aid the user in using the system. In this manner, valuable insight will be derived towards the development of geo-based information systems for remote sensing data in the EOS era.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of Dr. Tony Villasenor (NASA - GSFC), Dr. Chuck Klose and Mr. Iveory Johnson (NASA - JPL), and those numerous individuals who are providing us with much useful input and data. Research for this project was sponsored under NASA grant NAGW-987.

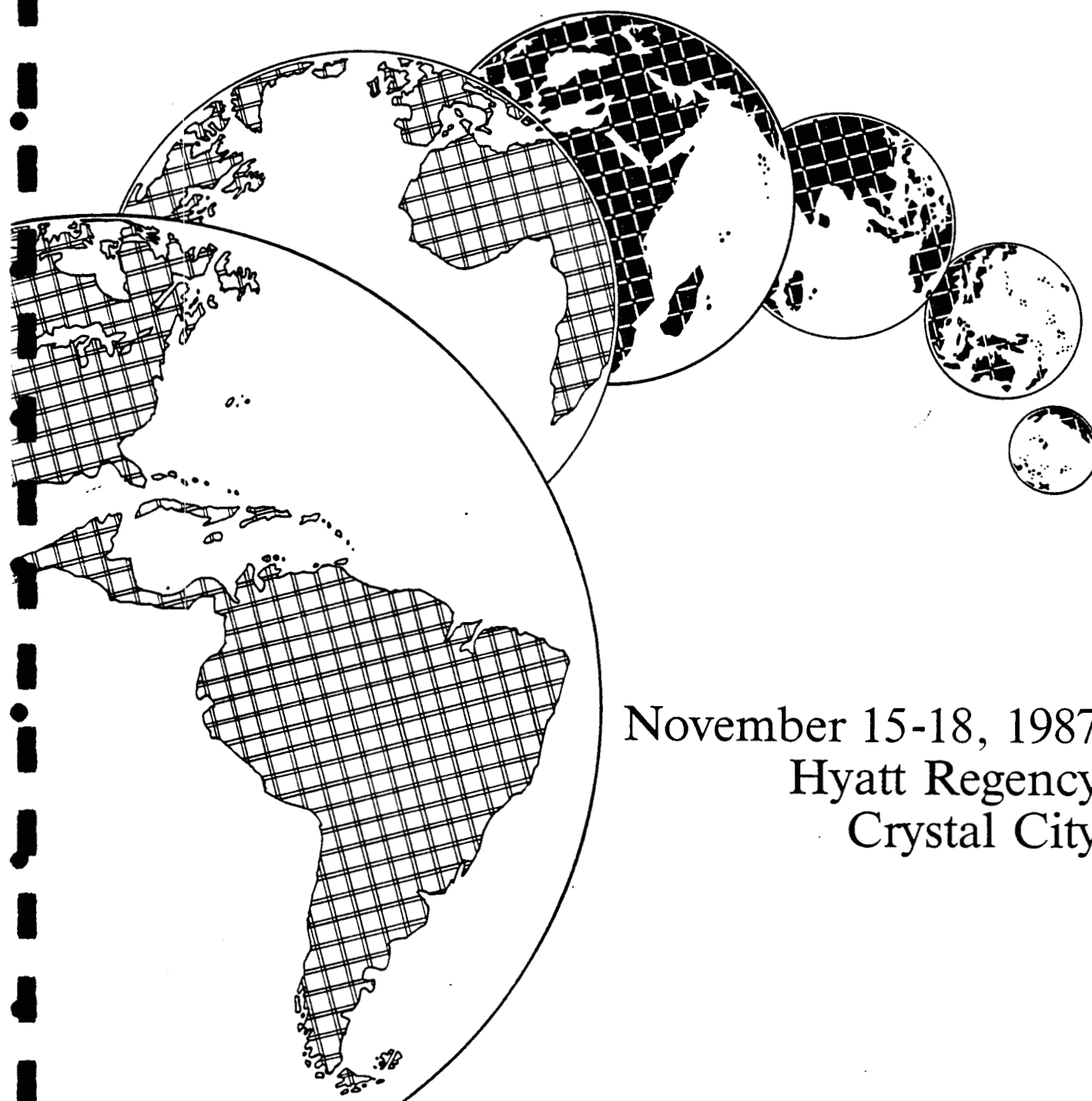
REFERENCES

- Abel, D.J. and J.L. Smith 1986. "A Relational GIS Database Accommodating Independent Partitioning of the Region", Proceedings of the Second International Symposium on

Spatial Data Handling, Seattle, Wa., pp. 213-224.

- Arvidson R.E., Butler D.M., and R.E. Hartle 1985. "Eos: The Earth Observing System of the 1990's", Proceedings of the IEEE, vol 73, no. 6, June 1985, pp 1025-1030.
- Estes J.E. and J.L. Star 1986. "Support for Global Science: Remote Sensing's Challenge", Geocarto International, vol 1, no. 1, pp 3-14.
- Lorie, R.A. and A. Meier 1984. "Using a Relational DBMS for Geographical Databases", Geo-Processing, vol 2, pp. 243-257.
- Palimaka, J., Halustchak, O., and W. Walker 1986. "Integration of a Spatial and Relational Database Within a Geographic Information System", Technical Papers 1986 ACSM-ASPRS Annual Convention, vol 3, Washington, D.C., pp. 131-140.
- Peuquet D.J. 1984. "A Conceptual Framework and Comparison of Spatial Data Models", Cartographica, vol. 21, no. 4, 1984 pp 66-113.

International Geographic Information Systems (IGIS) Symposium: The Research Agenda



November 15-18, 1987
Hyatt Regency
Crystal City

ELECTRONIC BROWSING FOR SUITABLE GIS DATA

Jeffrey L. Star, David M. Stoms, Mark A. Friedl, and John E. Estes

Remote Sensing Research Unit
Department of Geography
University of California
Santa Barbara, CA 93106 USA
(805) 961-3845

ABSTRACT

With the mass of remotely sensed data being collected, researchers are often unaware of what data is available and where it is stored. Data is frequently purchased on the basis of a terse database record describing, for instance, cloud cover and image quality. A better approach may be to allow analysts to preview spatial data, not just some salient facts about that data.

In a NASA-sponsored research program, the authors are developing a prototype system with browsing capability for data archives. The objective is to allow scientists, sitting at their local workstations, to access a network, to retrieve records of image and spatial data selected by user-specified attributes, to view low resolution versions of the data, and to place an order, if the data are satisfactory. The architecture and functioning of the BROWSE testbed is briefly outlined.

Surprisingly, the definition of BROWSE has found no consensus. This paper focuses on the on-going efforts to establish a definition that satisfies a broad range of scientific disciplines. Any operational definition will include the algorithms used to compress the raw data into a low resolution format. Preliminary algorithms being considered include single band subsetting, spatial subsampling, band ratios, principal components analysis, and linear combinations. Several browsing scenarios illustrate the complexities of selecting suitable data sets. An effective browsing utility will have benefits beyond NASA data systems and the Earth Observing System. Lessons learned from this project may be of value to other spatial data base designers.

1. INTRODUCTION

Effective information systems are an increasingly important aspect of spatial data analysis (Estes, 1985). Technical and social innovations provide a forcing function for improvements in the design of information systems, driven by the

increasing demand for and generation of scientific data. Naisbitt (1982) discussed several social transformations that characterize corresponding changes in how geographical analysis is being done. Among these trends are shifts from:

- centralization to decentralization
- hierarchies to networking
- an industrial to an information society and
- forced technology to high technology and 'high touch', meaning increased personal involvement.

The National Aeronautics and Space Administration (NASA) has recognized these trends in establishing pilot data systems for ocean, climate, land, and planetary science data (Jet Propulsion Laboratory, 1986a; NASA Goddard, 1986; NASA, 1986; Jet Propulsion Laboratory, 1986b). Working groups are also designing information systems for the vast quantity of data projected from the Earth Observing System (EOS) on the Space Station complex in the coming decade. The challenge is twofold. The first challenge lies in developing data handling techniques for the anticipated large data volumes. The second challenge is to bring this new technology to a broader scientific and applications constituency in the service of global science (Estes et al, 1986).

The Committee on Data Management and Computation (CODMAC) of the National Academy of Science predicts that an important capability in future spatial information systems will be the ability to browse through databases from a remote terminal (CODMAC, 1982). Browsing could perform three important functions. It would allow users to locate and preview spatial data and make an informed decision about their utility, for instance, as input to a geographic information system (GIS). Secondly, it would provide a mechanism for the user community to view newly received data and make recommendations on whether the data are suitable for permanent archiving. A third function of interactive browsing would be to aid decisions regarding what additional observations to acquire during an ongoing mission (CODMAC, 1986). At NASA, this mode of operation is being called *telescience*. However, different disciplines do not necessarily share common ideas on optimal browsing formats and characteristics. This paper describes attempts to define 'browsing' in the context of multidisciplinary spatial data systems. This research is part of an on-going project at the University of California, Santa Barbara Remote Sensing Research Unit, funded by NASA Headquarters.

A brief overview of the BROWSE testbed facility at UCSB is provided. (A more complete description can be found in Star et al, 1987). Next, some tentative algorithms for generating browseable data suitable to different disciplines are proposed. In addition, several scenarios for browsing, suggested by our scientific collaborators, are outlined to suggest the complexity of reaching a consensus in a definition. Then some future research directions are discussed. The main focus

of next year's effort will be in the evaluation phase. Our hope is that public discussion of these issues will not only improve the effectiveness of the BROWSE testbed, but will also stimulate creation of browsing utilities on other spatial information systems as well.

2. DESCRIPTION OF THE BROWSE TESTBED FACILITY

The first year of research, recently ended, saw the development of a rapid prototype of a browse utility at UCSB. Our objective was to get an operating version on-line and accessible via phone line to a set of collaborators from many disciplines and institutions around the United States. In the current year, we plan to incorporate their feedback on the usefulness of the system for their respective needs. What follows is a very brief description of the testbed facility.

The host facility consists of a MicroVAX II Workstation with peripheral tape drive, two 319 megabyte hard disks, a VT260 terminal, and a PC-AT as a workstation. Resident on the host computer are the database management system (DBMS), the user interface program, two databases, and a set of preprocessed browseable images. The DBMS used is a public-domain package, Relational Information Management (RIM), also being used by the NASA Ocean Data System (NODS) operated by the Jet Propulsion Laboratory.

To save the user from having to learn RIM syntax or the details of the databases, we have written a menu interface that prompts the user for query attributes. The Catalog data base contains detailed records of individual images. As seen in the Catalog menu in Figure 1, the user can restrict the search to specific geographic area, sensors, dates, maximum cloud cover, and browseable status. The graphic mode for identifying a geographic search space is shown in Figure 2, using Santa Barbara, California as an example. The program then automatically formulates the appropriate database query. The current query attributes or the current set of records retrieved can then be displayed. Figure 3 shows an example of records retrieved for the same area as shown in Figure 2. Realize that at this point in the session, only information about the data is provided, not imagery. This aspect is similar to a computerized literature search at a library or the type of query the EROS data center has supported for several years (USGS, 1980).

When a Catalog database record sounds useful, the user invokes KERMIT, a public domain communications package, to transmit the preprocessed image to their local terminal. This low resolution version can then be displayed locally using an image display program written for the IBM PC-AT workstation. Using multiple windows, the display includes a single band image, its histogram, and statistics. Currently, the display uses a level slicing routine to divide the grey values into eight colors, which can be adjusted through contrast enhancement functions in the program. Figure 4 shows a black and white reproduction of the BROWSE image corresponding to the database record of Figure 3. At this point,

UCSB-BROWSE	CATALOG Data Base Menu	Version 1.0	Revised 08/13/87
1. SELECT GEOGRAPHIC AREA 2. SELECT PLATFORM/SENSOR NAMES 3. SELECT DATE RANGE 4. SELECT MAXIMUM CLOUD COVER 5. SELECT ON-LINE STATUS 6. DISPLAY ATTRIBUTE CRITERIA 7. DISPLAY SELECTED RECORDS 8. DONE WITH ATTRIBUTE SELECTION			


Figure 1. Catalog Menu for BROWSE
Showing Query Attributes for Restricting Search

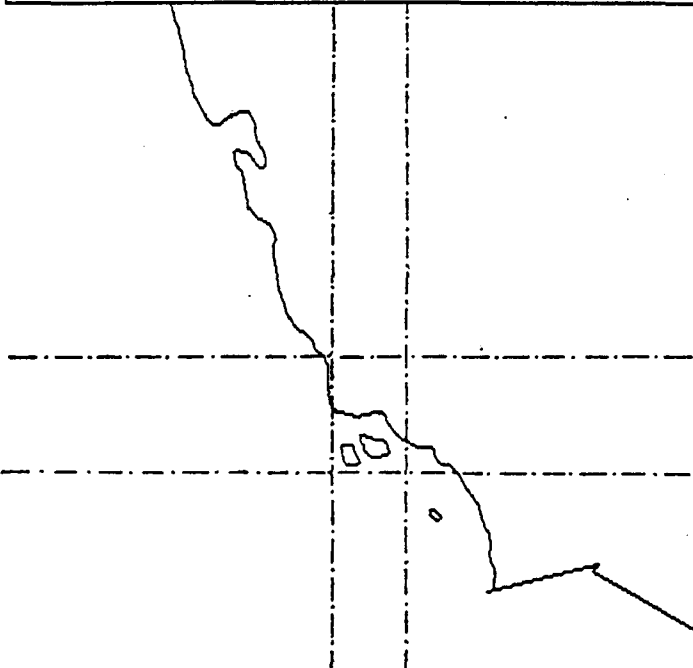
UCSB BROWSE - WORLD PLOT

USER MENU

1: ZOOM
2: UNZOOM
3: RECORD COORDINATES AND RETURN TO MAIN MENU

PLEASE MAKE A SELECTION:





DISPLAY INFORMATION

CONTINENT: NORTH AMERICA

REGION:

	MIN	MAX
LAT:	15.00	90.00
LON:	170.00	-45.00

ZOOMED COORDINATES

	MIN	MAX
LAT:	31.00	39.62
LON:	127.15	112.79

Figure 2. Example of Graphic Mode for Specifying
Geographic Region Around Santa Barbara, California

ORIGINAL PAGE IS
OF POOR QUALITY

UCSB-BROWSE: Current number of records selected = 3

#	SCENE ID	CLD%	DATE	PLATFORM	MISS	SENSOR	ON-LINE?
1)	Y5092817550X0PC2	20	09/15/86	LANDSAT	5	TM	ON
2)	Y5092817550X0BN3	20	09/15/86	LANDSAT	5	TM	ON
3)	Y5092817550X04/3	20	09/15/86	LANDSAT	5	TM	ON

End of current list of records

Would you like more details on any image? [Y/N]: y

Enter the number of the record you wish to see, between 1 and 3: 1

1) SCENE ID: Y5092817550X0PC2 DATE: 09/15/86 ON-LINE? ON
 PLATFORM: LANDSAT MISSION: 5 SENSOR: TM
 CLOUD COVER: 20% QUALITY: 5555555 HISTORY: PC2,SUBSET
 FORMAT: DIGITAL PHYSICAL DESC: NULL
 FILE POINTER: [.TM]SANTAB86.PC2

Figure 3. Example of Data Retrieved from the Catalog
for Santa Barbara, California

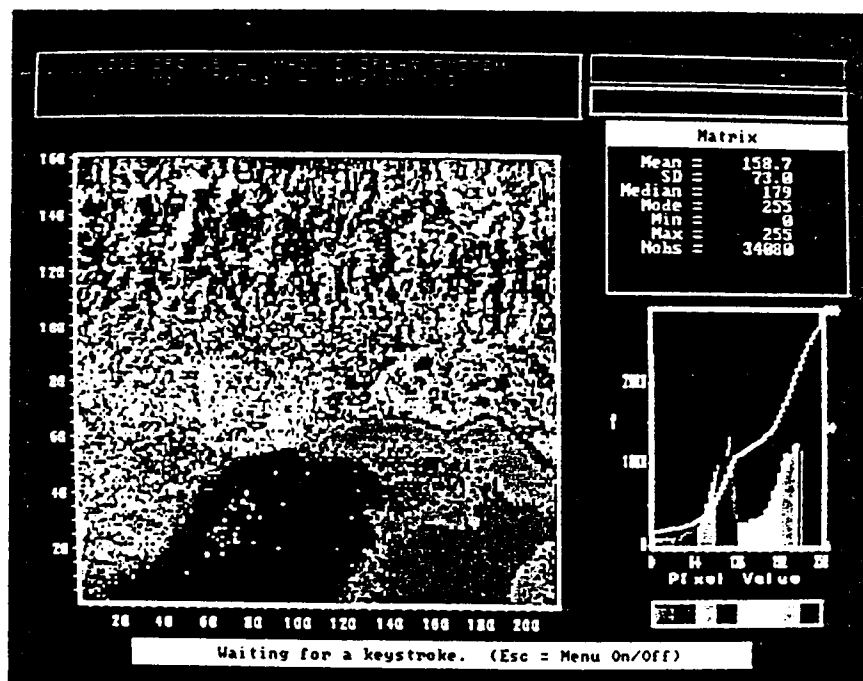


Figure 4. BROWSE Display of Scene of Santa Barbara, California
Corresponding to Record in Figure 3
(original display is in color)

the user can make a more informed decision on the suitability of the data for a spatial data processing application than from the data record alone. What makes BROWSE unique is the capability of both locating and displaying spatial data at a remote terminal.

An alternative menu path guides the user through the Directory database containing addresses and collection emphasis of various data centers around the world. Therefore, if suitable imagery is not available in the relatively small collection at the UCSB Remote Sensing Research Unit, the user can be directed to other likely sources. We plan to add Catalogs for some of these other archives to the BROWSE network. A third database of user information is planned for the coming year. This User database will permit a researcher to find other investigators who are knowledgeable about particular geographic regions, sensors, or disciplines.

3. PROPOSED BROWSING FORMATS

From the description of the testbed facility, browsing in this context can be very generally defined as locating spatial data and viewing it in some graphical form. This is not unlike how a researcher browses in a library--going to the shelf of relevant books, selecting some likely candidates, looking at the abstracts (i.e., a low resolution version of the entire text), rejecting some, finding new suggestions in the list of references, and often making serendipitous discoveries in material they were not originally aware of. It is this sense of discovery and exploration that can add Naisbitt's sense of 'high touch' to the high technology of computerized data retrieval.

Each of the NASA pilot data systems are implementing some form of browse capability. NODS includes a browse facility with data tabulations, graphics, and image formats. The user chooses a format and is presented with a list of names of browse files to view (Jet Propulsion Laboratory, 1986a). These files contain preprocessed examples of each format. Browse images of 4500 Coastal Zone Color Scanner and Advanced Very High Resolution Radiometer scenes for the West Coast Time Series have been recently put on line. At the Pilot Climate Data System, browsing includes looking at image data, but it also refers to looking at descriptions of sensors and climate parameters (NASA Goddard, 1986). Browsing or 'quick-look' support is also specified in the functional requirements for the Pilot Land Data System. Significantly, browsing is included under the data management requirements rather than those for analysis software (NASA, 1986). Recognition of this role underscores the point that browsing is a function to aid in data selection and management and should not supplant or duplicate GIS or image processing analytical functions. The Planetary Data System, is using videodisks for electronic browsing of planetary images (Jet Propulsion Laboratory, 1986b). A browse utility for EOS data is also specified (Broome, 1986).

Other data systems outside NASA are also providing browsing functions (Kubo, 1986). However, little detail of what these 'quick-look low resolution' views should show is described in any of these systems.

Among our objectives in the BROWSE project is to determine if there is a basic set of processing algorithms to generate suitable browse images. At the end of the project, we intend to provide NASA with a matrix of algorithms preferred by each discipline and suggest a small subset of these algorithms that present data in ways the majority of researchers require. The challenge is to satisfy scientists in fields as diverse as oceanography, forestry, and meteorology. The difficulty can be illustrated by considering clouds as viewed by a spaceborne multispectral scanner. To many specialists, clouds are a hindrance to information extraction. Therefore, extensive cloud cover is a negative attribute to be avoided or masked from the data. For some meteorological applications, however, it is precisely the clouds that are significant.

Providing low resolution browse data presumably involves compression in the spatial, spectral, radiometric, or temporal domains, i.e., subsampling the original data. Data compression offers several benefits in an information system and one major drawback. On the positive side, compression reduces the data storage and transmission loads on the system. Although storage capabilities are increasing (and costs decreasing) in the computer industry, the impact of EOS's proposed 262 channel High Resolution Imaging Spectrometer (HIRIS) will still be enormous in the foreseeable future (Broome, 1986). The projected data rate from HIRIS translates into one 6250 bpi density tape every second. The disadvantage of compression is that, after a certain level of reduction, the process is irreversible. Just as an abstract cannot be transformed into the original book, a highly compressed image cannot be converted by the user into the raw data. In other words, BROWSE only supplies the abstracts, as it were, so that the user can decide whether to buy the whole book.

The simplest approach to image compression is to discard all but a single spectral band. For purposes of browsing, a single band is often adequate for determining whether image data is suitable for input and analysis in a GIS. A visible wavelength band is appropriate for revealing the effects of clouds and atmospheric disturbance, for determining the presence of man-made objects, or for penetrating water surfaces. Other wavelengths have their own best uses.

A second approach capitalizes on the known relationships between spectral bands by using ratios of two bands. A ratio of reflective infrared over red wavelengths is commonly used in vegetative studies. The relative brightness is an indicator of biomass (Jackson, 1984). Geologists frequently use a mid infrared over near IR, a mid-IR over another mid-IR, or a red over blue band ratio to determine composition of surface materials (Crippen et al, in press).

Principal components analysis takes advantage of correlation in a dataset to transform it into a smaller number of dimensions. Significant scene variance may be conserved in a few synthetic output channels (i.e., components), while

uncorrelated noise is collected in lower order components that can be discarded in many circumstances. We hypothesize that a single component will show enough information for a browser. The disadvantage of principal components analysis is that it can be very time consuming to compute. The transformation coefficients must be recalculated for every image, and the components may be difficult to interpret since they are scene dependent.

Linear combinations are another way to transform image data, using a predefined set of coefficients. Each combination makes a specific rotation of data in spectral space to optimize a particular feature. Perhaps the best known of this type is the Tasseled Cap transformations for Landsat Multispectral Scanner data emphasizing soil brightness and vegetation greenness (Kauth et al, 1976). A similar transformation has been developed for Thematic Mapper data (Crist et al, 1984). In essence, the first transformed axis is an ordination from wet to dry soil, while the second axis indicates biomass as a distance from the bare soil baseline. Coefficients can be easily derived for other sensors but the interpretation of the axes may be unique for each type (Jackson, 1983).

4. SOME BROWSING SCENARIOS

It is impossible to predict all the ways researchers will want to use the BROWSE facility. The collaborators who have assisted us by describing their browsing needs tend to be experienced in remote sensing applications. The new constituency NASA hopes to reach is harder to identify, and new users may have difficulty articulating what they need, other than that the system be helpful and flexible. This section describes a few of the scenarios which our multidisciplinary collaborators have identified as the process they might use in gathering GIS data.

As we progress into the age of global science, researchers are collecting datasets for large regions at high resolution. An analyst working at this scale might need to see a low resolution, single band view of many adjacent scenes acquired in a particular season by the same sensor under relatively clear atmospheric conditions.

Research projects sometimes involve multistage sampling, with low resolution data of large areas and higher resolution for selected samples. A browse facility could determine if data of all required types were available and suitable at the study site. Alternatively, when multisensor data is required, the DBMS could identify sites with all the required types. By browsing through these sites, the analyst could select the most suitable for further investigation and commitment of staff and funds.

Discipline specialists new to remote sensing may not know the capabilities of different sensors. In an extreme example, an investigator wanting to census elk populations over a large region may believe that a single scene of a low

resolution sensor like the Advanced Very High Resolution Radiometer (AVHRR) would be adequate. A quick-look at a browse image should convince them immediately that AVHRR data is not appropriate for the task.

Misplacing research data over time as students come and go is an embarrassing fact of life in many academic departments. It is hopeless, as we ourselves have discovered, to go through a storeroom full of poorly labeled tapes, trying to find a valuable dataset someone in the past has painstakingly assembled. If a browse image of each dataset were available, along with a trail to the dataset itself, researchers might spend less time recreating data.

5. FUTURE DEVELOPMENT PLANS

The first year of development focused on rapid prototyping of the BROWSE testbed facility and exploring what 'browse' means to different disciplines. The existing prototype consists of a single node at UCSB with a relatively small database. In the coming grant year, we intend to expand the system to include at least one or two additional nodes with different types of image data. To accomplish this involves developing a high level activity manager that queries the appropriate databases around the network, transparently to the user, so a person does not need to learn multiple query languages.

The prototype facility has now become accessible via standard phone lines to our collaborators. We will be monitoring their use and incorporating their feedback on the user interface and the types of browse formats. In concert with this, we will be examining data compression in greater detail. Tradeoffs between storage overhead of preprocessed image data versus time delays of processing on demand will be assessed.

The existing prototype assumes the user can select appropriate sensors and will recognize when data are suitable. Since NASA hopes to establish a wider constituency for data acquired in space, more expert guidance will be necessary to aid new users in formulating appropriate queries. Development of an expert system is planned that can give intelligent advice on query attributes. The user need respond only with the parameters of their task i.e., the view that is more familiar to them. The initial expert enhancement will be in the domain of vegetation applications where a substantial knowledge base can be assembled from the research literature.

6. CONCLUSIONS

A capability to browse spatial data will be an important function in future spatial data systems. Researchers will browse in order to locate suitable input data, to recommend imagery for permanent archiving, and to request additional

data acquisitions from ongoing missions. When EOS begins generating massive volumes of data in support of global science, the need for these three functions will be even greater. The BROWSE project outlined here indicates one effort by NASA headquarters to prepare for the EOS era.

As defined above, 'browsing' involves both locating and viewing spatial data. The browse function is most appropriate as a data management tool. We do not see it as a replication of existing GIS or image processing analytical functions. Several methods of compressing data into browseable format have been discussed. Additional methods, from disciplines different from those of the authors, may be identified during the testing and evaluation phase. Planned enhancements should make BROWSE more effective in serving a wider constituency of users, which is one of NASA's objectives. To be effective, a browsing function should also be fast, flexible, and friendly.

Persons interested in joining the list of BROWSE users/testers are encouraged to contact the authors at the above address. Together we can give browsing for spatial data a degree of 'high touch' comparable to the level of high tech provided by computerized database management.

Acknowledgments

The authors would like to acknowledge the contributions of Dr. Tony Villaseñor and Dr. Jim Dodge at NASA Headquarters, Dr. Chuck Klose, Pat Hogan, and Ivory Johnson at the Jet Propulsion Laboratory, and those numerous individuals who are providing us with much useful input and data. Research for this project was sponsored under NASA grant NAGW-987.

References

- Broome, Jr., Douglas R., "The Earth Observing System: A Multidisciplinary System for the Long-Term Acquisition of Earth Science Data from Space," *Monitoring Earth's Ocean, Land, and Atmosphere from Space--Sensors, Systems, and Applications*, vol. 97, pp. 797-822, 1986.
- CODMAC, *Data Management and Computation*, 1: Issues and Recommendations, 167 pp., National Academy Press, Washington, D.C., 1982.
- CODMAC, *Issues and Recommendations Associated with Distributed Computation and Data Management Systems for the Space Sciences*, 111 pp., National Academy Press, Washington, D.C., 1986.
- Crippen, R.E., R.G. Blom, and J.H. Heyada, "Directed Band Ratioing for the Retention of Perceptually-Independent Topographic Expression in Chromaticity-Enhanced Imagery," *International Journal of Remote Sensing*, in press.

- Crist, Eric P. and Richard C. Cicone, "A Physically-Based Transformation of Thematic Mapper Data--The TM Tasseled Cap," *IEEE Transactions on Geoscience and Remote Sensing*, vol. GE-22, no. 3, pp. 256-263, 1984.
- Estes, John E., "The Need for Improved Information Systems," *Canadian Journal of Remote Sensing*, vol. 11, no. 2, pp. 124-131, 1985.
- Estes, J. E. and J. L. Star, "Support for Global Science: Remote Sensing's Challenge," *IEEE Proceedings*, vol. 73, no. 6, pp. 1097-1107, 1986.
- Jackson, Ray D., "Spectral Indices in n-Space," *Remote Sensing of Environment*, vol. 13, pp. 409-421, 1983.
- Jackson, Ray D., "Remote Sensing of Vegetation Characteristics for Farm Management," *Proceedings of SPIE*, vol. 475, pp. 81-96, 1984.
- Jet Propulsion Laboratory, *Software Specification for the GOLD Catalog*, 74 pp., California Institute of Technology, Pasadena, CA, 1986a.
- Jet Propulsion Laboratory, *Planetary Data System Version 1.0 System Specification*, 234 pp., California Institute of Technology, Pasadena, CA, 1986b.
- Kauth, R.J. and G.S.Thomas, "The Tasseled Cap--A Graphic Description of the Spectral-Temporal Development of Agricultural Crops as Seen By Landsat," *Symposium on Machine Processing of Remotely Sensed Data*, pp. 4B-41 through 4B-51, 1976.
- Kubo, Sachio, "The Basic Scheme of TRINITY: A GIS with Intelligence," *Proceedings of the Second International Symposium on Spatial Handling*, pp. 363-374, Seattle, WA, 1986.
- Naisbitt, J., *Megatrends*, 333 pp., Warner Books, New York, NY, 1982.
- NASA, *Pilot Land Data System Functional Requirements--Version 0.2*, 13 pp., , 1986.
- NASA Goddard Space Flight Center, *Proceedings of the Second Pilot Climate Data System Workshop*, NASA Conference Publication 2430, 263 pp., National Space Science Data Center, Goddard Space Flight Center, Greenbelt, MD, 1986.
- Star, Jeffrey L., Mark A. Friedl, David M. Stoms, and John E. Estes, "Browse Capability for Spatial Databases," *Proceedings of GIS'87 Conference*, San Francisco, CA, 1987.
- USGS, *A User's Guide to the Teletype-Inquiry Option of the INORAC System*, EROS Data Center, Sioux Falls, SD, 1980.

Biographical Sketches

Dr. Jeffrey L. Star is manager of the Remote Sensing Research Unit and Lecturer at the University of California, Santa Barbara. He received his undergraduate degree from M.I.T. and his doctorate in oceanography from Scripps Institution of Oceanography, University of California, San Diego. His research interests are in remote access to large archives of spatial data, merging artificial intelligence with geoprocessing technologies, and creation of a regional GIS for agricultural monitoring. Dr. Star has served on a number of national committees, including NASA's Earth Observing System Data Panel and the Task Force on Geo Information of the Research Libraries Group.

David M. Stoms is a Ph.D. student in the Department of Geography at the University of California, Santa Barbara. His Masters thesis was entitled 'Preliminary Design of a Farm Monitoring Geographic Information System'. Prior to graduate school, he worked 12 years with the U.S. Forest Service, primarily in land management planning. His current research interests are in browsing capabilities for spatial databases, knowledge-based approaches to spatial data processing and retrieval, and the value of improved spatial information.

Mark A. Friedl is a Masters student in the Department of Geography at the University of California, Santa Barbara. He received his B.S. from McGill University in Montreal in geography in 1986. While at McGill, he was involved with research at the McGill Subarctic Research Station and with the McGill Advanced Cartography Laboratory. He is currently involved in research related to spatial data handling and knowledge-based interpretation of remotely sensed imagery.

Dr. John E. Estes is Professor of Geography and Director of the Remote Sensing Research Unit at the University of California, Santa Barbara. He has authored more than 250 works on remote sensing and information systems, including 'Fundamentals of Image Analysis', a chapter in the *Manual of Remote Sensing*, which he coedited. He has served as senior scientist at the National Aeronautics and Space Administration and is a member of the Committee on Data Management and Computation of the National Academy of Science.

MAPWD: An Interactive Mapping Tool for Accessing Geo-Referenced Data sets

Mark A. Friedl, Kenneth C. McGwire and Jeffrey L. Star

Remote Sensing Research Unit, University of California
Santa Barbara, California 93106

ABSTRACT

Many scientists use geographic location as a fundamental attribute to identify interesting earth science data sets. The specification of this attribute in database queries is both tedious and prone to error. MAPWD is a graphical tool which simplifies this specification by allowing the user to point to areas of interest using an electronic map, instead of manually entering coordinate values. With this tool the user is able to identify regions of interest using maps with comparable detail at a wide range of spatial scales. This functionality has been achieved by hierarchically structuring the database of map vector data into separate files based on both geographic location and resolution.

Introduction

Earth science data sets are growing rapidly in both size and variety. Data archives are currently being compiled by numerous agencies at a wide range of spatial scales which are of a higher accuracy, resolution and volume than ever before. These data sets encompass a multiplicity of formats and data types including topographic, soils, climate, oceanographic, and hydrologic data to name but a few. An important issue confronting earth science is the efficient management of these scientific data sets.

The development of an efficient earth science data management system encompasses a diverse range of issues including:

- database design parameters
- optimal strategies for data archiving and deletion
- superior user access to distributed data archives
- conformal formats for data exchange and accuracy standards.

In this paper we describe a program called MAPWD. This program comprises one component of a user friendly interface to spatially referenced data cataloged in a relational database management system. The goal of this interface is to aid the user in specifying queries to the underlying database.

A fundamental requirement of queries to databases of spatially referenced data is the efficient specification of search constraints based upon geographic attributes. The purpose of MAPWD is to simplify the specification of locational attributes for queries to the database. This interactive spatial query system allows the user to specify a geographic region of interest by pointing to locations on a virtual map that is displayed on the workstation rather than manually entering the corresponding geographic coordinates.

MAPWD is written in Fortran77 using a Digital Equipment Corporation (DEC) implementation of the graphics kernel standard (GKS) graphics programming standard (level 1b) to implement the various graphics necessary for this type of utility (DEC 1987). The MAPWD program represents one component of a larger system entitled "BROWSE". BROWSE is designed as a testbed for implementing and evaluating strategies for improving user accessibility to data archives (Star et al., 1987). The BROWSE data management system for which MAPWD was developed is specifically designed to handle remotely sensed data. However, most of the concepts described in this paper may be generalized to any spatially referenced data set (i.e. maps, photography, and geophysical data).

Spatially Referenced Data Sets

Most earth sciences data sets may be considered to be a subset of the more general category of geographical data. Geographical data are characterized by both spatial and non-spatial attributes which may be used to locate objects stored within the data base. Spatial attributes describe characteristics such as the geographic coverage of the data, while non-spatial attributes may include items such as the date of data acquisition and other data set specific information. Geographical data sets are defined to be spatially referenced in that the location of each object in the database is included using some fixed geographic coordinate system. In addition, topological aspects (e.g., adjacency, proximity, etc.) as well as descriptive attribute information may be recorded for any object. Indeed, it is the topological and spatial characteristics of geographical data processing systems that distinguish them from more conventional data management systems for applications such as banking or library systems (Burrough 1986).

Database management systems (DBMS) have been the focus of extensive research and development in computer science. Current data models which have been developed for DBMS technology are highly sophisticated. However, the majority of this effort has been geared towards handling data which is non-spatial in nature. Consequently, in recent years there has been much interest among the geo-processing community in adapting existing DBMS technology to handle geographic data in a more natural manner (van Roessel 1987, CODMAC 1986, Lorie and Meir 1984). In particular, the relational data model (Codd 1970) has been employed for a variety of geographical data processing applications.

Browse

The BROWSE system (Star et al. 1987, 1988) is a prototype data management system designed to manage remotely sensed and other spatially referenced data sets. The system has been implemented on a MicroVAX II using a public domain relational DBMS called RIM (Relational Information Management, Johnson and Johnson 1985). This DBMS is equipped with a Fortran interface which we have used to develop a menu driven user interface to the underlying DBMS. The user interface is organized as a hierarchy of menus which guide the user's selection of query attributes. The user is prompted for attribute information, and the query is formulated and performed using subroutine calls embedded in the Fortran source code. In this manner, the user is not required to have a knowledge of relational algebra, nor of a specific query language.

A primary focus of BROWSE is related to examining techniques for improved user access to scientific data sets. Currently, numerous large scale scientific data sets are in various stages of creation, with spatial coverage varying in scope from regional to global (Mooneyhan, 1987). These data sets encompass numerous disciplines and are being archived at different locations using different digital storage systems and data formats.

A key to the effective utilization of these highly valuable data archives is maximizing the accessibility of the scientific community to these data. As part of the solution to this problem the Committee on Data Management and Computation (CODMAC) has recommended that:

"NASA evaluate and develop the concept of an 'electronic browse' capability, allowing users to explore data files via communication links. We recommend that 'quick-look' low resolution data be included for use in electronic browsing" (CODMAC 1982).

Capabilities such as those specified above are the focus of the BROWSE project. Running on workstations which emulate Tektroniks 4014 displays, the MAPWD routine embedded within the BROWSE system represents one specific case of improving the ability for researchers to efficiently access earth science databases. Other techniques for improving the utility and general applicability of the BROWSE system such as a quick look image display capabilities and object oriented search strategies have been developed or are in planning (Wright and Friedl 1987, Stoms et al. 1988).

MAPWD

A fundamental attribute of earth science data sets is location and/or areal coverage. MAPWD is a tool which allows users to specify geographical regions (windows) of interest in a database query by pointing to them on an electronic map displayed at their workstation. This program is implemented as part of the user interface to the BROWSE system. However, the general concepts could be easily modified and the source code adapted to any database of spatially referenced datasets. The program is written in DEC Fortran77 and requires a Tektroniks 4014 class graphics workstation, or one equipped with emulation software. A complete set of the DEC GKS run-time libraries (level 1b) is also necessary to run the software. A complete source code listing is provided in appendix A.

The user is able to iteratively refine the spatial window specification by successively zooming into and away from a region of interest. When the user is satisfied with the window selection, the geographic coordinates are passed to BROWSE as part of a query specification. We believe this to be a superior approach to manually typing in coordinate values using the keyboard. Using a conventional relational database system, a user would specify a geographic

region of interest by manually entering the coordinates for the window containing that region. For example, using a SQL (structured query language, Chamberlin et al. 1976) based query language the general form of this query would be:

```
SELECT sceneid FROM images
WHERE NW_lat lt 40
      and NW_long lt -120
      and SE_lat gt 30
      and SE_long gt -118
```

This query would find those scene identification numbers of images which lie within the geographic window with north west corner coordinates of 40N 120W and south east corner coordinates of 30N 118W. Specifying such a query is both tedious and prone to error because the user is forced to manually access atlas type products, estimate coordinate boundaries for study areas, translate these values into the particular coordinate system of the database, and finally include these values in a database query.

In contrast, MAPWD provides the BROWSE user with a simple and explicit method of selecting geographic areas of interest. The MAPWD display consists of four window areas (fig. 1):

- 1) a reference map window,
- 2) a zoomed map window,
- 3) a menu window,
- 4) and a display information window.

MAPWD is called from within BROWSE. To use the system, a user selects the menu item in BROWSE corresponding to "select geographic area." The user then identifies a general region of interest within a map of the globe and is

presented with a base map of that area. Using a mouse or keystrokes, the user defines a rectangular window of interest on the base map by pointing to the desired window corners. The zoomed display window is then rescaled and plotted with only those features located within the specified window. Features outside of that window are clipped (fig. 2). When the final area of interest has been defined, the coordinates are returned to the BROWSE main program and included as part of a relational database query.

The algorithm for efficiently performing this procedure consists of simple graphic manipulations using hierarchically organized data components. Vector files containing the map data at several levels of resolution (i.e., detail) are stored on-line in disk files. Based on the window specification a set of vector files is selected and plotted. The vector files are in latitude/longitude format and are stored in binary form to reduce space requirements and improve I/O performance.

The zoom mechanism which allows the user to iteratively refine a specific geographic window is simple to implement within a GKS graphics environment. Functions for both 'picking' map coordinates and automatic clipping are built into this graphics standard. A more difficult problem is to develop an efficient method for plotting the vector files at various levels of resolution. Stated in another way, it is necessary to have vector files at several levels of resolution which may or may not be plotted depending on the map scale to which the user is zoomed. Consider, for example, the case of North America. At the continent level it is both unnecessary and inefficient to plot U.S. county boundaries. At the same time, if the display is to provide useful geographic information for orientation, it is necessary to include this level of detail when examining regions with the areal extent of only a few states.

A solution to this problem was achieved by hierarchically structuring the

vector database into different files and assigning a zoom level to each file (fig. 3). In addition, at each zoom level the vector database is segmented geographically into separate files. Such segmentation of a cartographic database is referred to as "tiling" (Burrough 1986). The zoom level and geographic bounds for each vector file are stored in a lookup table for each continent. For any given window selection, a table lookup is performed and those vector files which are at the appropriate zoom level and which are within, or intersect, the geographic window selected are plotted. In addition, a point file of cities and major physiographic features is plotted in a similar manner.

In this way, the plotting procedure is greatly optimized compared to storing one high resolution vector file which is plotted at all scales. Also, this format allows the database of vector files to be dynamically updated without having to update the actual source code. Instead, vector files may be updated, added, or deleted by simply modifying the lookup table resident in a disk file. This concept may also be extended to include different types of thematic data at different levels of resolution depending on the specific scientific domain and geographic area of interest.

The alternative to this type of approach is to use line generalization techniques (Douglas and Peucker 1973). Line generalization algorithms modify the degree of precision in vector representation dynamically at run time. A simple example of this type of approach would be to systematically exclude points from the same vector file at successive zoom levels. However, this type of algorithm is less efficient, more difficult to implement, and prone to distortion effects (Monmonnier 1982) relative to the multi-resolution vector file approach implemented in MAPWD.

A key utility of the MAPWD approach to specifying geographic windows is that it is adaptable to searching for data sets at a wide range of space scales,

ranging from highly local to global in scope. For example, a micro-climatologist may require data for a region of less than 100 square kilometers. Concurrently, an investigator interested in earth system science investigations may wish to search for data on a continental or even global scale. In this context, MAPWD allows the user to specify windows using maps of comparable detail at effectively any resolution. The technique employed by MAPWD also allows for improved multidisciplinary access to a database, as lookup tables may selectively display vector files which contain information which is relevant to a specific field of study. For example, a researcher searching for data on regional vegetation will be presented with a map of physical and political features of continents, while an oceanographer would be presented with a map of bathymetric features.

Conclusions

The development of techniques for improved access to the vast collection of earth sciences data sets anticipated in the 1990's will be of great importance for advanced earth science research in the near future. While only part of a larger effort to develop these capabilities in the BROWSE program, the MAPWD spatial query system represents a valuable method of simplifying and improving the capability of identifying relevant earth science data sets. This is achieved by the interactive specification of geographic bounds using electronic maps at a variety of scales. The format of data storage and presentation within MAPWD allows for rapid and flexible spatial data querying for users from numerous backgrounds conducting studies at any number of scales. Any readers interested in using the BROWSE system and associated MAPWD software should contact any of the authors through the Department of Geography at UCSB.

Acknowledgments

Funds for this work were provided by NASA grants NAGW-987 and NAGW-455. We are also indebted to Larry Carver at the University of California, Santa Barbara Map and Imagery Lab (MIL) who allowed us to develop the initial system on the MIL MicroVax II.

References

- Burrough, P.A., 1986, Principles of geographical information systems for land resources assessment: Clarendon Press, Oxford, 193 p.
- Chamberlin, D.D., Astrahan, M.M., Eswaran, K.P., Griffiths, P.P., Lorie, R.A., Mehl, J.W., Reisner, P., and Wade, B.W., 1976, Sequel 2: a unified approach to data definition, manipulation, and control: IBM Journal of Research and Development, v. 20, no. 6, p. 560-575.
- Codd, E.F., 1970, A relational model for large shared data banks: Comm. of the ACM, v. 13, no. 6, p. 377-387.
- CODMAC, 1982, Data management and computation, 1. Issues and Recommendations: National Academy Press, Washington, D.C., 167 p.
- CODMAC, 1986, Issues and recommendations associated with distributed computation and data management systems for the space sciences: National Academy Press, Washington, D.C., 111 p.
- DEC, 1987, VAX GKS reference manual vol 1: Digital Equipment Corporation, AI-HW43B-TE, Maynard, Ma.
- Douglas, D.H., and Peucker, T.K., 1973, Algorithms for the reduction of the number points required represent a digitized line or its caricature: Canadian Cartographer, v. 10, no. 2, p. 112-122.
- Johnson, J., and Johnson, I., 1985, Relational information management (RIM) user's guide, Version 4.5: JPL-715-604, Jet Propulsion Laboratory, Pasadena, Ca., 56 p.
- Lorie, R.A., and Meir, A., 1984, Using a relational DBMS for geographical databases: Geo-Processing, v. 2, p. 243-257.
- Mooneyham, D.W., 1987, An overview of geographic information systems within the United Nations Environment Programme: Proceedings of GIS'87 Symposium, San Francisco, Ca., American Society of Photogrammetry, Falls Church, Va, p. 536-543.
- Monmonnier, M.S., 1982, Computer-assisted cartography principles and concepts: Prentice Hall, Inc., Englewood Cliffs, N.J. 214 p.
- Star, J.L., Friedl, M.A., Stoms, D.M., and Estes, J.E., 1987, Browse capability of spatial databases: Proceedings of GIS'87 Symposium, San Francisco, Ca., American Society of Photogrammetry, Falls Church, Va, p. 196-205.
- Star, J.L., Stoms, D.M., Friedl, M.A., and Estes, J.E., 1988, Electronic browsing suitable GIS data: Proceedings of the International Geographic Information Systems Symposium, Crystal City, Va., p. 321-332.
- Stoms, D.M., Star, J.L., and Estes, J.E., 1988, Knowledge-based image data management: an expert front end for the Browse Facility, Proceedings of the

ASPRS/ACSM Annual Convention, Vol. 4; St. Louis, Mo., American Society of Photogrammetry, Falls Church, Va, p. 69-78.

Wright, R.K., and Friedl, M.A., 1987, The development of a microcomputer based image processing system: Unpublished manuscript presented at the 1987 ASPRS/ACSM annual convention, Baltimore, Md.

van Roessel, J.W., 1987, Design of a spatial data structure using relational norms: International Journal of Geographical Information Systems v . 1, no. 1, p. 33-50.

Captions for Figures

Figure 1. MAPWD display layout consisting of multiple windows. Clockwise from upper left: 1. menu window; 2. reference map window; 3. display information window; 4. zoomed map window.

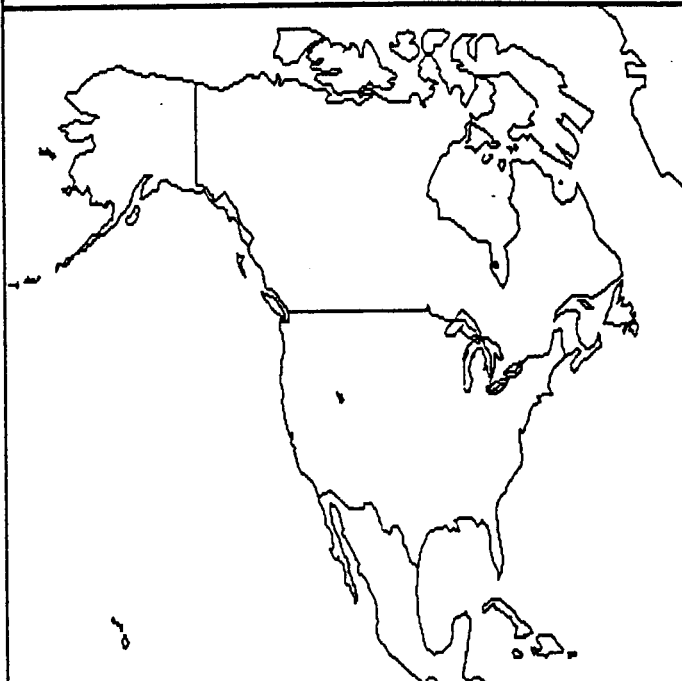
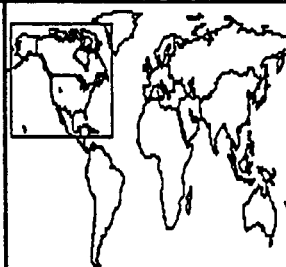
Figure 2. Sequence of map zooming operations to specify geographic window in southern California. Note variable resolution and changing degree of detail as iterative zooming is performed.

Figure 3. Schematic representing hierarchical structuring of vector database. Multiple vector files are stored with variable degrees of resolution or detail. At any specific zoom level, vectors are segmented geographically using a technique known as tiling. A lookup table is used to plot appropriate vector files based on the zoom level and window of the area selected by the user.

UC68 BROWSE - WORLD PLOT

USER MENU

- 1: ZOOM
 - 2: UNZOOM
 - 3: GO BACK TO LAST ZOOM
 - 4: RETURN WINDOW TO BROWSE
- PLEASE MAKE A SELECTION:



DISPLAY INFORMATION

REFERENCE MAP:

WORLD3.OUT

MIN MAX

LAT: -70.00 90.00

LON: -180.00 180.00

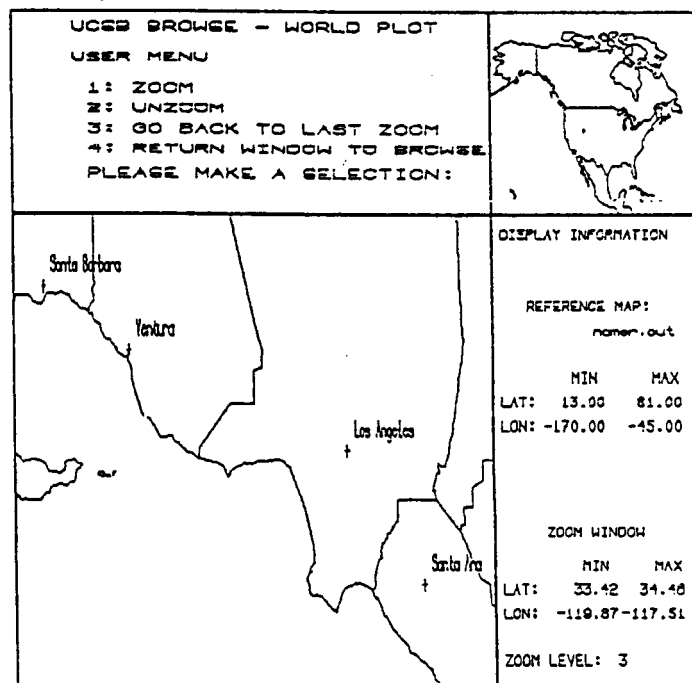
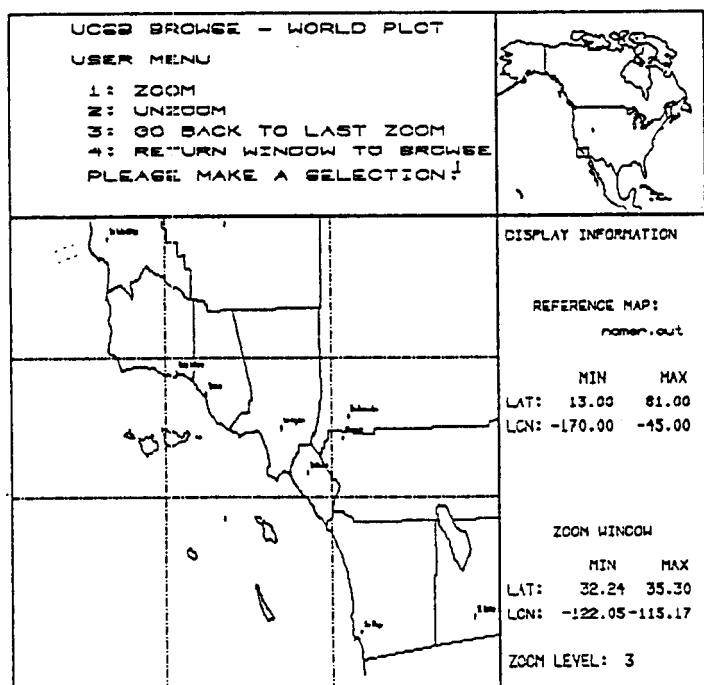
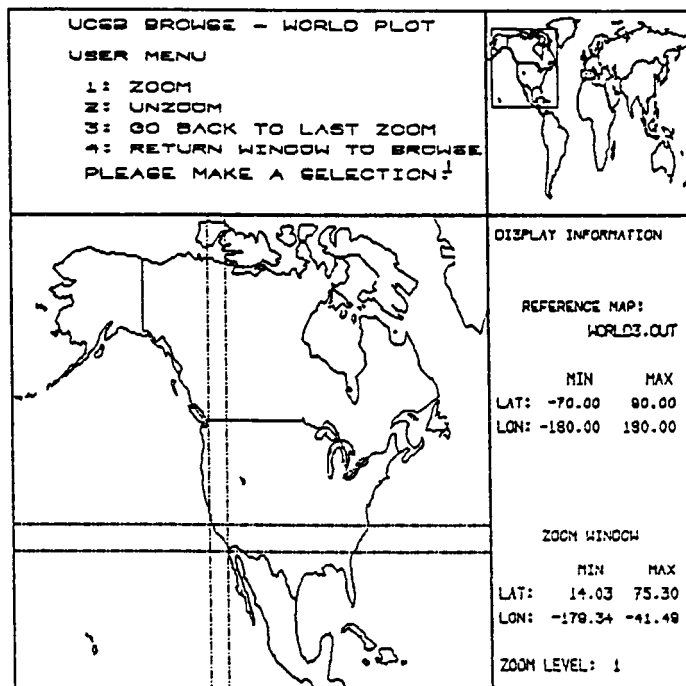
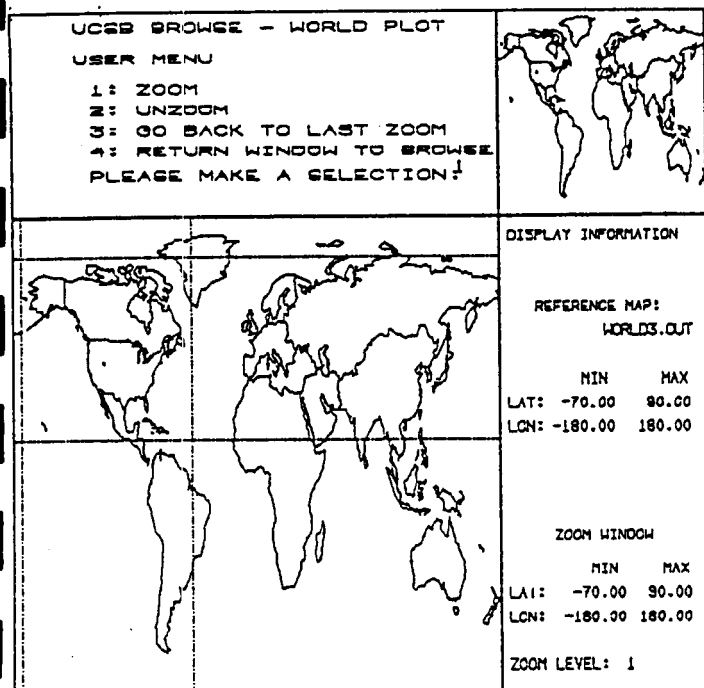
ZOOM WINDOW

MIN MAX

LAT: 15.19 78.48

LON: -179.34 -41.49

ZOOM LEVEL: 1



ORIGINAL PAGE IS
OF POOR QUALITY

ORIGINAL PAGE IS
OF POOR QUALITY



LEVEL 1 - GLOBAL

CONTINENTS

EUROPE
AFRICA
NORTH AMERICA
SOUTH AMERICA
AUSTRALIA
ASIA



LAT	LON
38.0000	120.0000
38.0000	100.0000
27.0000	27.0000
27.0000	27.0000
6.0714	6.0714
6.0714	6.0714
34.0000	34.0000
34.0000	34.0000

LEVEL 2 - REGIONAL

COUNTRIES
STATES

CANADA
UNITED STATES
MEXICO
GUATEMALA
COSTA RICA
...

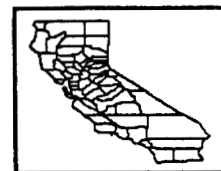


LAT	LON
38.0000	120.0000
38.0000	100.0000
27.0000	27.0000
27.0000	27.0000
6.0714	6.0714
6.0714	6.0714
34.0000	34.0000
34.0000	34.0000

LEVEL 3 - LOCAL

COUNTIES

SANTA BARBARA
VENTURA
KERN
SAN DIEGO
LOS ANGELES
...



LAT	LON
38.0000	120.0000
38.0000	100.0000
27.0000	27.0000
27.0000	27.0000
6.0714	6.0714
6.0714	6.0714
34.0000	34.0000
34.0000	34.0000

IV. Appendices

Source Code

```

      subroutine banner(menuid)
c*****
c  Banner Display Subroutine
c  reads menu00.txt file for menuid and prints to terminal  *
c
c  author: david stoms
c  date : 11/24/87
c  revised: 06/30/88
c*****
c declarations
      integer i,nlines,ios
      character menuid*10,string*78,title*30,filename*22
      call clearsreen
c open menu text file and read text
      filename(1:7) = ['.text',
      filename(8:17) = menuid
      open(unit = 7,file = filename,status = 'old',err = 9005)
c read header and continue
      read (7,9011) string
      read (7,9012) nlines
c print menu to terminal
      5 do 50 i = 1, nlines
      read (7,9011) string
      print 9001,string
      50 continue
      close(unit=7)
      100 return
9001 format(' ',1X,A78)
9005 print 9006,ios,filename
9006 format(' ',1X,'-ERROR- ',12,
      2 : Can not open ',A22,' file.')
9011 format(A78)
9012 format(12)
9999 end
      subroutine bantek
c*****
c  Browse Bantek Subroutine
c  called by many banner subroutine
c  displays graphic banner in tek 4010 mode
c
c  author: david stoms
c  date : 03/14/88
c  revised: 08/30/88
c*****
c declarations
      include 'sys$library:gsdefs.bnd'
      character*1 answer
      real x(9),y(9)
      call gswm(1.0,0.1,0.0,0.1,0)

```

```

      call gsvp(1.0,1.0,0.95,0.2,0.75)
      call gselnt(1)
c outline coordinates in world coords
      x(1) = 0.0
      x(2) = 0.0
      x(3) = 1.0
      x(4) = 1.0
      x(5) = 0.0
      x(6) = 0.0
      x(7) = 1.0
      x(8) = 1.0
      x(9) = 0.0
      y(1) = 0.0
      y(2) = 1.0
      y(3) = 1.0
      y(4) = 0.0
      y(5) = 0.0
      y(6) = 0.25
      y(7) = 0.25
      y(8) = 0.85
      y(9) = 0.85
c goto heavy line width for border
      call gslwsc(4.0)
      call gpl(9,x,y)
c switch to stroke precision for tall letters
      call gstxtp(2,2)
      call gk$set text height(0.2)
      call gtx(0.1,0.5,'BROWSE')
      call gtx(0.101,0.50,'BROWSE')
      call gtx(0.102,0.50,'BROWSE')
      call gtx(0.103,0.50,'BROWSE')
      call gtx(0.097,0.50,'BROWSE')
      call gtx(0.098,0.50,'BROWSE')
      call gtx(0.099,0.50,'BROWSE')
      call gtx(0.1,0.501,'BROWSE')
      call gtx(0.1,0.502,'BROWSE')
      call gtx(0.1,0.503,'BROWSE')
      call gtx(0.1,0.497,'BROWSE')
      call gtx(0.1,0.498,'BROWSE')
      call gtx(0.1,0.499,'BROWSE')
c print rest of banner message in small text height
      call gk$set text height(0.025)
      call gtx(0.1,0.9,'Welcome to the NASA BROWSE Testbed')
      call gtx(0.1,0.16,'University of California, Santa Barbara')
      call gtx(0.1,0.13,'Remote Sensing Research Unit MicroVAX II')
      call gtx(0.1,0.08,'As of 09/01/88: 68 images on-line')
      call gtx(0.6,0.0,'')
c return to regular line width
      call gslwsc(1.0)
      return
      end
      subroutine user(menuid)
c*****
c  Browse User Subroutine
c  called from query subroutine with menu23.txt
c  calls menu.sub for user menu, reads choice
c  for either database,schema,help,mail,return to main, or quit *
c

```

```

c      author: david stoms
c      date   : 5/29/87
c *****
c declarations
c
c      integer option
c      character menuid*10
c
c      print menu to terminal
c
c          call menu(menuid)
c
c          10 print 1000
c          1000 format(' ', 'Select Option #: ')
c          read *, option
c          if (option .lt. 1 .or. option .gt. 6) then
c              print 9000
c          9000 format(' ', 'option must be between 1 and 6')
c          print 9000
c          goto 10
c      endif
c
c      goto(100,200,300,400,500,600),option
c
c      query user data base option
c
c          100 print *, 'welcome to browse user DATA BASE'
c          call usrdbrq
c          goto 9999
c
c      leave goto statements in or program drops thru to other
c      menu choices
c
c      query user schema option
c
c          200 print *, 'welcome to browse user SCHEMA'
c          call usrsqc
c          goto 9999
c
c      query user schema option
c
c          300 print *, 'watch this space for future options'
c          goto 9999
c
c          400 print *, 'welcome to help'
c          goto 9999
c
c      return to main menu
c
c          500 menuid = 'menu01.txt'
c          call mainmenu(menuid)
c
c          600 call quit
c
c          9999 return
c          end
c
c      subroutine catalog(menuid)
c *****
c      Browse Catalog (INVENTORY) Subroutine
c      -called from query subroutine
c      -using menu21.txt
c      -prompts user for values for inventory database query
c      including location range, platform, sensor,
c      cloud cover and range of dates
c      -calls rim subroutines to retrieve tuples
c *****

```

```

c      author: david stoms
c      date   : 01/22/88
c      revised: 07/21/88
c *****
c declarations
c
c      include '[stoms.src]params.blk'
c
c      common /term/ termtype
c      integer choice, termtype
c      parameter (NATFS = 8)
c      character menuid*10, errmsg*74, answer*1
c      real*8 newrel, oldrel
c      real*8 dbname(3) /8H$disk2:[,8H$STOMS]CA,8HT /
c      real*8 dbname(3) /8H$INVENT ,
c          2      8H
c          3      8H
c          ;
c
c      call rimstrnt
c      call clearscreen
c      print *, '... opening INVENTORY data base...'
c      call rimopn(cdbname)
c      if (rimstat .ne. 0) then
c          call rimmsg(errmsg)
c          print *, errmsg
c          end if
c
c      remove temp relations just in case of a previous crash
c      call cleanup
c
c      first ask what parameters they want to search on *****
c
c          1 call menu(menuid)
c          call getchoice(choice)
c
c      check for legal values
c      if ( choice .lt. 1 .or. choice .gt. NATFS) then
c          print 9200,NATFS
c          call pause
c          goto 1
c      end if
c
c      call clearscreen
c      goto (300,100,400,500,550,600,700,9999),choice
c
c      get location range *****
c
c      if user has vt100 emulator, don't allow graphic mode choice
c      300 continue
c      if ((termtype .eq. 1) .or. (termtype .eq. 3)) goto 320
c      FOLLOWING GAVE USER CHOICE OF GRAPHICS OR KEYBOARD IN TEK MODE
c      print 9300
c      print 9070
c      call getchoice(choice)
c
c      allows pointing to map for location range
c      if (choice .eq. 1) then
c          write(20,*) ' Using mapwd for lat/long'
c          call mapwd
c          goto 1
c      end if
c
c      get location coords from keyboard *****
c      320 write(20,*) ' Using keyboard for lat/long'
c      call listloc

```

```

      goto 1
c get platform type *****
100 call listplat
   if (numtuples .eq. 0) goto 1
   call clearsreen
c get sensor type *****
200 call listsens
   goto 1
c get time range *****
400 menuid = 'menu35.txt'
   call listtime(menuid)
   menuid = 'menu21.txt'
   goto 1
c get maximum cloud cover desired *****
500 call listcloud
   goto 1
c get on-line status *****
550 call listline
   goto 1
c display criteria *****
600 write(20,*) '      Display attribute criteria'
   call dispatts
   goto 1
c display selected records *****
700 continue
   write(20,*) '      Display selected records'
c find current name of temp relation
   call rselcst(ptr-1,newrel,oldrel,attrib,kalen)
   call dispans(newrel)
   menuid = 'menu21.txt'
   goto 1
c format statements *****
9001 format(' ', 'Do you wish to erase all temporary'
2      ' relations? [Y/N]: ', $)
9002 format(A1)
9003 format(' ', 'Enter choice number: ', $)
9100 format(' ', '-----')
9200 format(' ', 'Choice must be between 1 and ', 12, '/')
c 9300 format(' ', 'Do you wish to use: ', //)
c 2      12X, '1) GRAPHICS MODE--Point to area on a map', //
c 3      12X, '2) KEYBOARD MODE--Type in lat/long range', //)
9999 continue
c remove all temp relations and then close database
   call cleanup
9990 ptr = 0
   call rimcld
   if (rimstat .ne. 0) then
       call rimmsg(ermmsg)
       print *, ermmsg

```

```

   end if
   return
end
subroutine cleanup
*****
c Browse Cleanup Subroutine
c called by catalog subroutines
c calls rimrem for all temporary relations created during
c session; initially it asks user if they want to do so
c
c author: david stoms
c date : 11/23/87
c revised: 05/11/88
*****
c declarations
common /rimcom/ rimstat, unit, status
   real*8 tempname(12)
   integer ptr, rimstat, unit, status
   character*1 answer
c initialize tempname array
   tempname(1) = 6HTEMP01
   tempname(2) = 6HTEMP02
   tempname(3) = 6HTEMP03
   tempname(4) = 6HTEMP04
   tempname(5) = 6HTEMP05
   tempname(6) = 6HTEMP06
   tempname(7) = 6HTEMP07
   tempname(8) = 6HTEMP08
   tempname(9) = 6HTEMP09
   tempname(10) = 6HTEMP10
   tempname(11) = 6HTEMP11
   tempname(12) = 6HTEMP12
c ask if user wants to remove temp relations
c
c print 9001
c read (*, 9002) answer
c if(answer .eq. 'n') goto 9999
c do 10 ptr = 1, 12
c   call rimrem(tempname(ptr))
c   if (rimstat .eq. 2) then
c       print *, '...cleaning is done...'
c   call pause
c   goto 9999
c end if
c print 9003, tempname(ptr)
c continue
10
c format statements *****
c9001 format(' ', 'Do you wish to erase all temporary'
c9002 ' relations? [Y/N]: ', $)
c9003 format(A1)
c9003 format(' ', '...removing ', A6, ' relation...')
9999 return
end
subroutine clearsreen
*****
c Browse Clearsreen Subroutine
c called by many browse subroutines

```

```

c clears screen, cursor to upper left, for next whatever *
c uses gks functions for tek mode *
c
c author: david stoms *
c date : 01/22/88 *
c *****
c declarations
c
c include 'sys$library:gksdefs.bnd'
c
c common /term/ termtype
c integer termtype
c character*5 clearscrn(3)
c character*7 home(3)
c
c goto (100,200,300) termtype
c
c char(27) == esc ... prepending a blank to the control strings for safety
c codes for vt200 series
100 continue
c data clearscrn(1) ' [2J',home(1) ' [1;1H' /
c clearscrn(1)(2:2) = char(27)
c home(1)(2:2) = char(27)
c write (*,9001) clearscrn(termtype)
c goto 9999
c
c code for tek 4010/4014
200 continue
c gks opened in setterm subroutine
c
c call gclrwk(1,0)
c call gtx(0.1,0.99, ' ')
c
c continue
c goto 9999
c
c just print 5 blank lines for plain ascii terminals
300 print 9010
c
c code for ibm pc fortran
c data clearscrn(3) 'Z'201b5b324a' /,home(3) ' /
c
c 9001 format(1a5)
c 9002 format(1a7,$)
c 9010 format(' ',//)
c 9999 return
c end
c
c *****
c Browse Dataprompt Program
c
c Prompts operator for catalog data fields for BROWSE Testbed *
c and writes records to image.cat and loc.cat files *
c
c author: david stoms *
c date : 02/10/88 *
c revision date : 02/22/88 *
c *****
c declarations
c common /term/termtype

```

```

integer termtype(1) /
integer cloudcvt, year, month, day
real cent_lat, cent_lon
c character sceneid*16, physdesc*16, quality*8, history*16,
2 online*8, format*8, sensor*8, platform*8, mission*4,
3 pointer*20, geodesc*16, gridx*8, gridy*8, today*9,
4 operator*20, slash*1, answer*1
c
c variable definitions *****
c
c sceneid = accession number unique to each scene
c cloudcvt = percent cloud cover in image
c year = 2 digit number of year of image acquisition
c month = 2 digit number of month of image acquisition
c day = 2 digit number of day of image acquisition
c physdesc = descriptor of main physical feature in scene
c quality = descriptor of image quality
c history = processing history of data, eg. subset, pc1
c online = is a browse file available [y/n]?
c format = film, digital, paper product
c sensor = sensor name/acronym
c platform = platform name/acronym
c mission = mission number for given platform
c pointer = pathname of browse file or physical storage location
c of original data
c geodesc = geographic place name of key location in image
c gridx = path name/number, if any for given sensor system
c gridy = row name/number, if any for given sensor system
c cent_lat = latitude of center point of image; + = north, - = south
c cent_lon = longitude of center point of image; + = east, - = west
c
c today = today's date for identification of input
c operator = name of person entering data for identification of input
c
c open image.cat and loc.cat files *****
c
c open(unit=7, file = 'loc.cat', access = 'append',
2 status = 'unknown', err = 9910)
c open(unit=8, file = 'image.cat', access = 'append',
2 status = 'unknown', err = 9910)
c write(7,9501)
c write(8,9502)
c
c get date and operator name
c call date(today)
c print 9001
c print 9030
110 print 9002
c read (*,9021) operator
c if(operator .eq. ' ') then
c print 9004
c goto 110
c end if
c call UPCASE(operator)
c
c write operator and date to data files for identification
c write(7,9503) operator, today
c write(8,9503) operator, today
c
c begin data entry *****
100 continue
c call clearscreen
120 print 9001
c print 9031
c print 9003
c read (*,9009) sceneid

```

```

call UPCASE(sceneid)

print 9005
read (*,9006) cloucdvr      ladd check for value 0-100
print 9007
read(*,9008)month,slash,day,slash,year ladd check of data values

print 9010
read (*,9009) physdesc
call UPCASE(physdesc)

print 9011
read (*,9012) quality
call UPCASE(quality)

print 9013
read (*,9009) history
call UPCASE(history)

print 9014
read (*,9012) online
call UPCASE(online)

print 9015
read (*,9012) format
call UPCASE(format)

print 9016
read (*,9012) sensor
call UPCASE(sensor)

print 9017
read (*,9012) platform
call UPCASE(platform)

print 9018
read (*,9019) mission
call UPCASE(mission)

print 9020
read (*,9021) pointer
call UPCASE(pointer)

print 9022
read (*,9009) geodesc
call UPCASE(geodesc)

print 9023
read (*,9012) gridx
call UPCASE(gridx)

print 9024
read (*,9012) gridy
call UPCASE(gridy)

print 9025
read (*,9026) cent_lat
print 9027
read (*,9026) cent_lon

c write records to files
write(7,9201)geodesc,gridx,gridy,cent_lat,cent_lon,sceneid
write(8,9200)sceneid,cloucdvr,month,day,year,
2 physdesc,quality,history,online
write(8,9209)format,sensor,platform,mission,pointer

```

```

c do you want to enter another record?
print 9100
read (*,9101) answer
if((answer .eq. 'Y') .or. (answer .eq. 'y')) goto 100
c quit
goto 9999

c format statements *****
9001 format(' ',14X,'---UCSB BROWSE DATA ENTRY PROGRAM---')
9002 format(' ',5X,'What is your name? ',5X)
9003 format(' ',5X,'Scene ID # [16 Characters] > ',5X)
9004 format(' ',8X,'You must enter your name to proceed. ')
9005 format(' ',5X,'% Cloud Cover [Integer] > ',5X)
9006 format(13)
9007 format(' ',5X,'Date [MM-DD-YY] > ',5X)
9008 format(12.2,A1,12.2,A1,12.2)
9009 format(A16)
9010 format(' ',5X,'Physical Description [16 Characters] > ',5X)
9011 format(' ',5X,'Data Quality [8 Characters] > ',5X)
9012 format(A8)
9013 format(' ',5X,'Processing History [16 Characters] > ',5X)
9014 format(' ',5X,'On Line Status [OFF or ON] > ',5X)
9015 format(' ',5X,'Data Format [FILM,PAPER,or DIGITAL] > ',5X)
9016 format(' ',5X,'Sensor [8 Characters] > ',5X)
9017 format(' ',5X,'Platform [8 Characters] > ',5X)
9018 format(' ',5X,'Mission # [4 Characters] > ',5X)
9019 format(A4)
9020 format(' ',5X,'Pointer [20 Characters] > ',5X)
9021 format(A20)
9022 format(' ',5X,'Place Name [16 Characters] > ',5X)
9023 format(' ',5X,'Path or Column [8 Characters] > ',5X)
9024 format(' ',5X,'Row [8 Characters] > ',5X)
9025 format(' ',5X,'Center Point Latitude [+/-XX.XXX] > ',5X)
9026 format(F8.3)
9027 format(' ',5X,'Center Point Longitude [+/-XXX.XXX] > ',5X)
9030 format(' ',20X,'Written by David Stoms',//,
2 23X,'revised 02/22/88',//)
9031 format(' ',//,5X,'Enter data at the prompts. Use the data',
2 type shown in brackets. '//)

9100 format(' ',//,5X,'Do you want to enter another record? [Y/N]: ',5X)
9101 format(A1)
9125 format(' ',//,' -ERROR- Cannot open data files')

9200 format(' ',16,A16,16,2X,13,2X,12.2,2X,12.2,2X,12.2,
2 2X,16,A16,16,5X,16,A16,16,2X,16,A16,16,2X,16,
3 AB,16,3X,16)
9209 format(' ',5X,16,A8,16,2X,16,A8,16,2X,16,A8,16,2X,
2 16,A4,16,2X,16,A20,16)
9201 format(' ',16,A16,16,2X,16,A8,16,2X,16,A8,16,
2 2X,SP,F7.3,2X,F8.3,2X,16,A16,16)

9501 format(' ',*(data input for locator relation in catalog)',/,
2 LOAD LOCATOR')
9502 format(' ',*(data input for image relation in catalog)',/,
2 LOAD IMAGE')
9503 format(' ',*(Input by: ',A20,5X,'Date = ',A9,')',/)

c error routine
9910 print 9125
stop

9999 write(7,*) 'END'

```

```

write(8,*) 'END'
close(unit=7)
close(unit=8)
end

```

```

c*****

```

```

subroutine UPCASE(TEXT)

```

```

c converts text to uppercase for input to RIM data files
c so that RIM doesn't get confused over case changes
c otherwise certain images may not be selected by query

```

```

c declarations

```

```

character*20 TEXT
integer length

```

```

c print 9001,TEXT

```

```

length = len(TEXT)

```

```

do 100 I=1,length

```

```

c if not lower case, do nothing

```

```

2 if ((ichar(TEXT(I:1)) .lt. 97) .or.
1 (ichar(TEXT(I:1)) .gt. 122)) then

```

```

c if lower case, convert

```

```

else TEXT(I:1) = char(ichar(TEXT(I:1)) - 32)

```

```

end if

```

```

100 continue

```

```

c print 9001,TEXT

```

```

c call pause

```

```

c9001 format(' ',5x,'text = ',A)

```

```

9999 return

```

```

end

```

```

subroutine directory(menuid)

```

```

c*****

```

```

c Browse Directory Subroutine

```

```

c -called from query subroutine with menu22.txt

```

```

c -prompts user for values for directory database query

```

```

c including geographic coverage, platform, sensor,

```

```

c format, and discipline emphasis

```

```

c -calls rim subroutines to retrieve tuples

```

```

c author: david stoms

```

```

c date : 01/13/88

```

```

c revised: 07/21/88

```

```

c*****

```

```

c declarations

```

```

include '[stoms.src]dir.blk'

```

```

integer choice

```

```

parameter (NATTS = 7)

```

```

character menuid*10,errmsg*74

```

```

real*8 newrel,oldrel

```

```

c real*8 dbname(3) /8H$DISK2:[,

```

```

c 2 8H$STOMS1D1,

```

```

c 3 8H

```

```

c real*8 dbname(3) /8HDIR

```

```

c 2 8H

```

```

3

```

```

8H

```

```

/

```

```

call rimstrt
call clearscreen
print *, '...opening DIRECTORY data base...'
call rimopen(dbname)
if (rimstat.ne. 0) then
call rimmsg(errmsg)
print *,errmsg
end if

```

```

c remove temp relations just in case of a previous crash
call cleanup

```

```

c first ask what parameters they want to search on *****

```

```

1 call menu(menuid)
call getchoice(choice)

```

```

c check for legal values

```

```

if (choice.lt. 1 .or. choice .gt. NATTS) then
print 9200,NATTS

```

```

call pause

```

```

goto 1

```

```

end if

```

```

call clearscreen

```

```

goto (100,300,500,550,600,700,9999),choice

```

```

c get platform type *****

```

```

100 call lisplat2

```

```

if(numtuples .eq. 0) goto 1

```

```

call clearscreen

```

```

c get sensor type *****

```

```

200 call lissens2

```

```

goto 1

```

```

c get desired geographic coverage *****

```

```

300 call listcont

```

```

call clearscreen

```

```

c if 'All' chosen, don't ask for countries

```

```

if(continent(1) .eq. 3HALL) goto 1

```

```

c now ask for country in chosen continent *****

```

```

350 call listcount

```

```

goto 1

```

```

c get discipline emphasis *****

```

```

c 400 call listdisc(discipline)

```

```

c print 9100

```

```

c if 'All' chosen, don't ask for specializations

```

```

c if(discipline .eq. 'All') goto 1

```

```

c now ask for specialization *****

```

```

c 450 call listspec(discipline,special)

```

```

c goto 1

```

```

c get desired format *****

```

```

500 call listform
    goto 1
c get desired node *****
550 call listnode
    goto 1
c display selected criteria *****
600 continue
    write(20,*) '      Display attribute criteria'
    call dispatt2
    goto 1
c display selected rim records
700 continue
    write(20,*) '      Display selected records'
    call relselct2(ptr-1,newrel,oldrel,attrib,kalen)
    call dispans2(newrel)
    goto 1
c format statements *****
c 9070 format(' ', 'Enter choice #: ', $)
9100 format(' ', '-----', '/')
9200 format(' ', 'Choice must be between 1 and ', 12, '/')
9999 continue
    call cleanup
    ptr = 0
    call rimclo
    if (rimstat.ne. 0) then
        call rimmsg(errmsg)
        print *,errmsg
    end if
    return
end
subroutine dispans(relname)
c*****
c Browse Dispsans Subroutine
c called by catalog subroutine
c retrieves tuples from relname relation and displays them
c on the user's terminal
c
c author: david stoms
c date : 12/10/87
c revision date : 07/21/88
c*****
c declarations
include 'lstoms.src'params.blk'
common /rimcom/ rimstat,unit,status
integer rimstat,unit,status
real*8 relname,sortlist(8),attname,operator
real*8 locrel,locatt,locop,locval(2)
integer sortlen,value,ctr,record
character*74 errmsg
character answer*2

```

```

integer tuple(32),locrec(14)
real*8 sceneid(2),platname,mission,sensname,on_line
2 physdesc(2),quality,history(2),format,pointer(3)
real*8 geodesc(2),gridx,gridy,scenenum(2)
real cent lat,cent lon
integer cloudpct,year,month,day
equivalence(tuple(1),sceneid(1))
equivalence(tuple(5),cloudpct)
equivalence(tuple(6),year)
equivalence(tuple(7),month)
equivalence(tuple(8),day)
equivalence(tuple(9),physdesc(1))
equivalence(tuple(13),quality)
equivalence(tuple(15),history(1))
equivalence(tuple(19),on_line)
equivalence(tuple(21),format)
equivalence(tuple(23),sensname)
equivalence(tuple(25),platname)
equivalence(tuple(27),mission)
equivalence(tuple(28),pointer(1))
equivalence(locrec(1),geodesc(1))
equivalence(locrec(5),gridx)
equivalence(locrec(7),gridy)
equivalence(locrec(9),cent lat)
equivalence(locrec(10),cent lon)
equivalence(locrec(11),scenenum(1))
c if pointer still at 0, i.e. no restrictions on full data base,
c do not display
if (ptr.eq. 0) then
    print *, '... no records have been retrieved yet ....'
    call pause
    goto 9999
end if
c get tuples from relname relation and store in variables *****
call rimlock(relname,1,50)
if (rimstat.ne. 0) then
    call rimmsg(errmsg)
    print *,errmsg
end if
c dummy where clause that selects all records
attname = 4*YEAR
operator = 2HGE
value = 0
call rimfind(relname,attname,operator,value,1)
call dispsort(sortlist,sortlen)
c if no sortlen is zero, don't sort
if(sortlen.eq. 0) then
    goto 100
end if
print *, '...sorting data...'
call rimsort(sortlist,sortlen)
c print headings and records one screenful at a time
100 ctr = 1
do 20 j=1,1000
    call clearsreen
    print 9001,numtuples
    print 9002
    print 9005
    do 10 i=1,15

```



```

print 9021,numtuples
read(*,9010)record
c reset file pointer to first tuple
call rimfind(relname,atname,operator,value,1)
call rlsort(sortlist,sortlen)
do 320 k = 1,record
    call ringet(relname,tuple,1)
320 continue
c now get details from locator relation too
locrel = 7HLOCATOR
locop = 8HSCENENUM
locop = 2HEQ
local(1) = sceneid(1)
local(2) = sceneid(2)
call rimlock(locrel,1,50)
call rimfind(locrel,locop,local,2)
call ringet(locrel,lococ,2)

print 9022,record,sceneid(1),sceneid(2),month,day,year,
physdesc(1),physdesc(2),platname,history(1),history(2),
mission,quality,sensname,painter(1),pointer(2),
pointer(3),cloudpkt,format_on_line
print 9023,cent_lat,gridx,cent_lon,gridy,geodesc(1),geodesc(2)
c save record in external file if desired
print 9008
read (*,9009) answer
if ((answer.eq. 'n') .or. (answer.eq. 'N')) then
    goto 340
else
    write(20,*),' Details written to temp.dat file'
end if
open(unit = 7, file = 'temp.dat',access = 'append',
status = 'unknown',err = 9910)
write(7,9022)record,sceneid(1),sceneid(2),month,day,year,
physdesc(1),physdesc(2),platname,history(1),history(2),
mission,quality,sensname,painter(1),pointer(2),
pointer(3),cloudpkt,format_on_line
write(7,9023)cent_lat,gridx,cent_lon,gridy,geodesc(1),geodesc(2)
print 9031
goto 340
continue

c ask if browse file should be kermited to terminal
if(on_line .eq. 2HON) call kernspwn(pointer)

c ask if another record desired
print 9024
read(*,9009) answer
if ((answer.eq.'n').or.(answer.eq.'N')) then
    close(unit=7)
    goto 9990
end if
goto 310

c format statements *****
9001      format(' ',4X,'UCS-BROWSE: Current number of records ',
           'selected = ',I4/)
9002      format(' ',1X,' #',2X,'PLATFORM',1X,' SENSOR ',4X,'DATE',4X,
           'CLOCK',5X,'PROCESSING',5X,'ION?')
9003      format(' ',1X,I3,',') 1X,A8,1X,A8,2X,I2.2,'/',12.2,'/',12.2,
           '2X,I3,3X,A8,2X,A3)
9005      format(' ',1X,'-----',1X,'-----',1X,'-----',2X,
           '-----',2X,'-----',2X,'-----',/,)
9006      format(' ',//,' End of current list of records')
9008      format(' ',//,' Would you like to save this record in ',

```

```

2      'a disk file? [Y/N]: ', $)
9009 format(A)
9010 format(14)
9020 format(' ', //, ' Would you like more details on any image?',
2      '[Y/N]: ', $)
9021      format(' ', //, ' Enter the number of the record you wish'
2      ' to see, between 1 and ', 14, ': ', $)
9022 format(' ', //, 1X, 13, ', ', 2X, 'SCENE ID: ', 2A8, 12X, 'DATE: ', 12.2,
2      ' 12.2, ', //, 12.2, ', ')
3      7X, 'PHYSICAL DESC: ', 2A8, 7X, 'PLATFORM: ', A8, //
4      7X, 'HISTORY: ', A8, 21X, 'SENSOR: ', A6, //
5      7X, 'QUALITY: ', A8, 21X, 'MISSON: ', A6, //
6      7X, 'FILE POINTER: ', 2A8, A4, 4X, 'CLOUD COVER: ', 13, 'X', //,
7      7X, 'FORMAT: ', A8, 22X, 'ON-LINE? ', A3, //
9023 format(' ', 6X, 'CENTER LATITUDE : ', SP, F11.6, 9X, 'X-GRID: ', A8, //,
9      7X, 'CENTER LONGITUDE: ', SP, F11.6, 9X, 'Y-GRID: ', A8, //,
1     7X, 'PLACE NAME: ', 2A8, //,
2     7X, 'UCSB-BROWSE://')
9024 format(' ', //, ' Would you like another record? [Y/N]: ', $)
9025 format(' ', //, ' -ERROR- Cannot open TEMP.DAT file')
9030      format(' ', //, ' Type <RETURN> to continue; q to quit ',
2      'display: ', $)
9031 format(' ', //, 5X, 'To read this file, enter TYPE TEMP.DAT when!', //
2      ' 5X, 'you return to the BROWSE> prompt.', )
9910 print 9025
9990      call rimunkl(rename,1)
9999      return
       end
       subroutine dispans2(relname)
c*****
c Browse Dispans2 Subroutine
c called by directory subroutine
c retrieves tuples from relname relation and displays them
c   on the user's terminal
c author: david stoms
c date : 11/06/87
c revised: 02/10/88
c*****
c declarations
include 'lstoms.src\dir.blk'
common /rimcom/ rimstat,unit,status
integer rimstat,unit,status
real*8 relname,sortlist(8),attname,operator,noderel,
2      nodeatt,nodeop
integer sortlen,value,ctr,record,nodeval
character*74 errmsg
character*1 answer
integer tuple(21),noderec(33)
real*8 platname,sensname,online,nodename,facname,
2      formtype,ctrname(2),countname(2),state(2),
3      add1(3),add2(3),add3(3),add4(3),phone(2)
integer nodeid,nodenum
equivalence(tuple(1),nodeid)
equivalence(tuple(2),nodename)
equivalence(tuple(4),platname)
equivalence(tuple(6),sensname)

```

```

equivalence(tuple(8),formtype)
equivalence(tuple(10),countname(1))
equivalence(tuple(14),countname(1))
equivalence(tuple(18),state(1))

equivalence(noderec(1),facname)
equivalence(noderec(3),nodenum)
equivalence(noderec(4),online)
equivalence(noderec(6),add1(1))
equivalence(noderec(12),add2(1))
equivalence(noderec(18),add3(1))
equivalence(noderec(24),add4(1))
equivalence(noderec(30),phone(1))

c get tuples from relname relation and store in variables *****
    call rimlock(relname,1,50)
    if (rimstat.ne.0) then
        call rimmsg(errmsg)
        print *,errmsg
    end if

c dummy where clause that selects all records
    attname = 6HNODEID
    operator = 2HGE
    value = 0
    call rimfind(relname,attname,operator,value,1)
    c need a new sort routine for DIRECTORY
    c call dispsort(sortlist,sortlen)

c if no sortlen is zero, don't sort
c   if(sortlen.eq.0) then
c       goto 100
c   end if
c   print *, '...sorting data...'
c   call rimsort(sortlist,sortlen)

c print headings and records one screenful at a time
100  ctr = 1
    do 20 j=1,1000
        call clearscreen
        print 9001,numtuples
        c need to add feature to print page number = numpage
        print 9002
        print 9005
        do 10 i=1,15
            call rimget(relname,tuple,1)
        200  call rimstat and print error message
            if (rimstat.gt.0) then
                print *, '...rimstat after rimget = ',rimstat
            goto 9990
        end if
        if (rimstat.ne.-1) then
            if (print 9003,ctr,nodename,platname,sensname,formtype,
                2      countname(1),countname(2),state
                ctr = ctr + 1
            else
                print 9006
                goto 300
            endif
        10  continue
        call pause
        20  continue

c ask if user wants more details from data base

```

```

300  print 9020
    read(*,9009) answer
    if ((answer.eq.'n') .or. (answer.eq.'N')) then
        goto 9990
    else
        write(20,*) ' More details requested'
    end if
310  print 9021,numtuples
    read*,9010)record
    c reset file pointer to first tuple
    call rimfind(relname,attname,operator,value,1)
    do 320 k = 1,record
        call rimget(relname,tuple,1)
320  continue
    c now get details from NODE relation too
    noderec = 4HNODE
    nodeatt = 6HNODEID
    nodeop = 2HEQ
    c set value equal to nodeid i.e., tuple(1)
    nodeval = nodeid
    call rimlock(noderec,1,50)
    call rimfind(noderec,nodeatt,nodeop,nodeval,2)
    c call rimmsg(errmsg)
    c print *,errmsg
    c call rimget(noderec,2)
    c call rimmsg(errmsg)
    c print *,errmsg
    print 9022,record,platname,sensname,formtype,countname(1),
        2      countname(2),countname(1),countname(2),state(1),
        3      state(2),nodename,online,
        4      add1(1),add1(2),add1(3),add2(1),add2(2),add2(3),
        5      add3(1),add3(2),add3(3),add4(1),add4(2),add4(3),
        6      phone(1),phone(2)
    c save record in external file if desired
    print 9008
    read(*,9009),answer
    if ((answer.eq.'n') .or. (answer.eq.'N')) then
        goto 340
    else
        write(20,*) ' Details written to dtemp.dat'
    end if
    open(unit = 7, file = 'dtemp.dat', access = 'append',
        2      status = 'unknown', err = 9910)
    write(7,9022),record,platname,sensname,formtype,countname(1),
        2      countname(2),countname(1),countname(2),state(1),
        3      state(2),nodename,online,
        4      add1(1),add1(2),add1(3),add2(1),add2(2),add2(3),
        5      add3(1),add3(2),add3(3),add4(1),add4(2),add4(3),
        6      phone(1),phone(2)
    print *, '...RECORD SAVED IN DTEMP.DAT FILE...'
340  continue
    print 9023
    read(*,9009) answer
    if ((answer.eq.'n') .or. (answer.eq.'N')) then
        close(unit=7)
        goto 9990
    end if
    goto 310

c format statements *****
9001  format(' ',4X,'UCSB-BROWSE: Current number of records ',
        2      'selected = ',14/)
9002  format(' ',1X,' # ',2X,'NODENAME',2X,'PLATFORM',1X,'SENSOR',

```



```

c first ask what parameters they want to search on *****
i = 1
print 9001
read (*,9002) answer
if ((answer .eq. 'n') .or. (answer .eq. 'W')) then
    write(20,*) 'Records not sorted'
    goto 9999
end if
menuid = 'menu41.txt'
1 call menu(menuid)
call getchoice(choice)

c check for legal values
if ( choice .lt. 1 .or. choice .gt. NATTS) then
    print 9200,NATTS
    call pause
    goto 1
end if
call clearscreen
goto (100,200,300,400,500,600,9999),choice

c set sortlist *****
100 sortlist(1) = 8HPLATFORM
    write(20,*) 'Sorted on Platform'
    goto 1000

200 sortlist(1) = 6HSENSOR
    write(20,*) 'Sorted on Sensor'
    goto 1000

300 sortlist(1) = 4HYEAR
    sortlist(1 + 1) = 4HMONTH
    sortlist(1 + 2) = 3HDAY
    i = 1 + 2
    write(20,*) 'Sorted on Date'
    goto 1000

400 sortlist(1) = 4HCLDX
    write(20,*) 'Sorted on Cloud Cover'
    goto 1000

500 sortlist(1) = 7HHISTORY
    write(20,*) 'Sorted on Processing History'
    goto 1000

600 sortlist(1) = 6HONLINE
    write(20,*) 'Sorted on On-Line Status'
    goto 1000

1000 i = i + 1
    goto 1

c format statements *****
9001 format(' ',5X,'Do you wish to sort the output? [Y/N]: ',)
9002 format(A)
c 9070 format(' ',) Enter choice #: ' $)
9200 format(' ',//,' Choice must be between 1 and ',12,/)

c end message *****
9999 continue
    sortlen = i - 1
    return

```

```

end
subroutine getchoice(choice)
c*****
c Browse Getchoice Subroutine
c called by many browse subroutines
c gets character input from user, checks that it is a number
c and returns the integer value as the choice
c
c author: david stoms
c date : 02/09/88
c*****
c declarations
    character*5 answer
    integer choice,i,k,l,array(5),digit
    choice = 0
    l = 5
    90 write (*,9001)
    read (*,9002) answer
    c check that answer is really a number
    do 10 i = 1,5
        array(i) = ichar(answer(i:l))
        print 9004,array(i)
        if(array(i) .eq. 32) then
            l = i - 1
            goto 15
        else if ((array(i) .lt. 48) .or.
            (array(i) .gt. 57)) then
            2 print 9201
              choice = -1
              goto 9999
            end if
        10 continue
    c compute integer value of character answer
    15 print 9005,l
      do 20 k = 1,l
        digit = array(k) - 48
        print 9006,digit
        choice = choice + (digit * 10 ** (l-k))
      20 continue
    c print 9003,choice
c format statements *****
9001 format(' ',5X,'Enter choice number: ',)
9002 format(A)
9003 format(' ',5X,'Choice = ',l)
9004 format(' ',l,' Array(l) = ',l)
9005 format(' ',l,' l = ',l)
9006 format(' ',5X,'Digit = ',l)
9201 format(' ',) Your answer must be integer.'
9999 return
end

```

SUBROUTINE HELP

C*****jls April VMS revision 30 Sept 87, from 13 May, 1987
 C revised by stoms 10/1/87 for installation in browse

```

c purpose:
c take fixed-record-length file, of specified format, and
c the pre-built index file for it.
c specified input text file format:
c line 1: header, date of revision
c line 2: keyword
c line 3 -> (n-1): description
c line n: "n"
c line n+1: next keyword ...
c index file format:
c format (a40,1x,i4) - keyword, record
c note that the keyword file does not have to be alphabetized,
c or in any order at all, really.
c*****
c implicit integer (a-z)
c for now, permitting 400 key words of 40 characters max
c common /vers/ version,revdate
c character*40 keys(400)
c integer*2 pointer(400)
c integer*2 recnum, maxrec, helpnum
c integer ios
c character*80 tstring
c character revdate*8,version*3
c real version,/1.0/
c ver is the version number
c call clearsreen
c write(20,*) ' Help Menu'
c print 9100, version
c open the needed files LUN 1 for input, LUN 2 for output
c
c call openfile()
c read and print the header on the input file
c read (unit=1, fmt=9200, rec=1, err=9600) tstring
c write (*,*) i file header:
c write (*,*) tstring
c write (*,*) 'only dealing with first 400 keywords'
c read the keywords in.
c
c call loadkeys(keys, pointer, maxrec)
c write (*,*) maxrec, ' keywords found and installed.'
c main display and prompt loop
c
100 call display (keys,maxrec,helpnum)
c call clearsreen
c if(helpnum .eq. 0) return
c call gethelp (pointer(helpnum))
c call clearsreen
c goto 100
c formats
c
9100 format (' UCSB-BROWSE HELP, Version: 'A3,/)
c + ' using a fixed-length-record file, and the '
c + ' accompanying index file, ' for a random-access',
c + ' help system for the BROWSE project. JLS/UCSB.'///,
c
9200 format (a)
9250 format (a40,1x,i4)
c error traps
c
9600 write (*,*) ' error reading text file.'
c goto 9900
9500 write (*,*) 'error writing index file.'
9900 write (*,*) iostat = ' ios
c return
c END

```

```

c *****
c subroutine openfile()
c prompt for, and open the input file as unit #1 - fixed org
c prompt for, and create the output index file as unit #2 - random org
c both files opened READONLY.
c integer ios
c open (unit=1, file='f.text\help.txt', status='old',
c + form='formatted', access='direct', readonly,
c + recl=80, err=1050,iostat=ios)
c open (unit=2, file='f.text\helpi.txt', status='old', err=1000,
c + readonly, iostat=ios)
c return
c error conditions and printouts
c
1000 write (*,*) ' error opening the index file.'
c goto 1075
1050 write (*,*) ' error opening the help file.'
1075 write (*,*) ' iostat = 'ios
c write (*,*) ' error in subroutine openfile.'
c return
c formats
c
c *****
c subroutine loadkeys(keys, pointer, maxrec)
c load keywords and pointers from files into arrays
c
c character*40 keys(*)
c integer*2 pointer(*)
c integer*2 recnum, maxrec
c do 25 recnum=1,100
c read (unit=2, fmt=9250, err=30, end=30)
c + keys(recnum), pointer(recnum)
25 continue
c fall out of loop ==> end of file found
c
30 continue
c maximum number of records is 1 less than recnum right now:
c
maxrec = recnum-1
9250 format (a40,1x,i4)
c return
c end
c *****
c subroutine display (keys,maxrec,helpnum)
c display the list of keywords, select one for display
c
c for now, display is lousy if there are more than 45 key words
c character*40 keys(*)
c integer*2 maxrec, helpnum, i
75 do 100 i=1,maxrec,2
100 write (*,9000) i,keys(i),i+1,keys(i+1)
c write (*,9002)
9002 format(' ', 'Please choose a keyword by number ... 0=exit : ', $)
c ---->
c call getchoice(helpnum)
c
c test for out-of-range response
c if ((helpnum .LT. 0) .or. (helpnum .GT. maxrec)) then
c write (*,*) '*****'

```

```

c      write (.*,*) 'value out of range, please try again!'
c      write (.*,*) 'Choice must be between 0 and i,maxrec
c      write (.*,*) '*****'
c      call pause
c      call clearsreen
c      goto 75
endif
c test for end of routine ???
if (helpnum.lt. 1) then
  close (1)
  close (2)
else write(20,9001)keys(helpnum)
endif
return
9000 format (1x,i3,'-',i3,' | ',i3,'-',i3,a33)
9001 format (' ',7x,'Help requested on ',A40)
end

c *****
c      subroutine gethelp (recnum)
c      get and display the appropriate help message, based on
c      records starting at helpnum
c      character*80 tstring
c      character*1 dummy
c      integer*2 recnum
c      read (unit=1, fmt=9200, rec=recnum, err=9600) tstring
c      write (.*,*) tstring
1000 recnum = recnum + 1
c      read (unit=1, fmt=9200, rec=recnum, err=9600) tstring
c      if (tstring(1:1).eq. '-') then
c        call pause
c        return
c      else
c        write (.*,*) tstring(1:79)
c        goto 1000
c      endif
9600 write (.*,*) 'trouble reading the text file.'
c      call pause
c      return
9200 format (a)
end

c      subroutine htoc(pointer,chrpoint)
c *****
c      Browse HTOC Subroutine
c      converts Hollerith (real*8) file pointer name to
c      character string
c      character*8 pointer
c      author: david stons
c      date : 12/10/87
c *****
c      declarations
c      real*8 pointer(3)
c      character*24 chrpoint
c      print 9001,pointer(1),pointer(2),pointer(3)
c      decode(24,9000,pointer(1)) chrpoint
c      print 9002,chrpoint

```

```

c format statements *****
9000 format(A24)
c9001 format(' ',5x,'pointer = ',3A8)
c9002 format(' ',5x,'chrpoint in htoc = ',A24)
9999 return
end

c      subroutine kernspan(pointer)
c *****
c      Browse Kernspan Subroutine
c      asks user if they want to transfer a browse file to
c      their terminal, and spawns appropriate subprocess if yes *
c      based on connection type--modem, SPAN, or ARPANET *
c      *
c      author: david stons *
c      date : 01/08/88 *
c      revised: 10/04/88 *
c *****
c      declarations
c      common modetype
c      character constring*56,chrpoint*24,answer*1,string*78
c      character path*24,filename*24,dirstring*70
c      integer status,(16$spawn,dirlen,filestrt,modetype,pathlen
c      real*8 pointer(3)
c      convert hollerith pointer to character string
c      call htoc(pointer,chrpoint)
c      read path name from text file
c      open(unit=18,file='l.text\path.txt',status='old',err=9950)
c      read(18,9021)string
c      read(18,9022)pathlen
c      read(18,9021)path
c      close(unit = 18)
c      build command string to be spawned
c      do 10 i = 1,24
c        if(chrpoint(i:1).eq. 'J') then
c          dirlen = 1
c          filestrt = 1 + 1
c        else
c          goto 20
c        end if
c      continue
20 path(pathlen + 1:pathlen + dirlen) = chrpoint(2:dirlen)
c      filename = chrpoint(filestrt:24)
c      inquire what connection mode is used--dialup,decnet/span,arpanet
c      don't inquire once type is known
c      if (modetype .eq. 0) call transmode(modetype)
c      use appropriate transfer method-- 1)kermit, 2)copy, 3)ftp
c      if(modetype .eq. 1) then
c        constring(1:6) = 'acom1'
c        constring(7:pathlen+dirlen+6) = path
c        constring(pathlen+dirlen+7:56) = filename
c      else if (modetype .eq. 2) then
c        constring(1:5) = 'copy'
c        constring(6:pathlen+dirlen+5) = path
c        constring(pathlen+dirlen+5:56) = filename
c      else
c        constring(1:6) = 'acom3'
c        constring(7:pathlen+dirlen+6) = path
c        constring(pathlen+dirlen+6:56) = filename

```

```

end if

c find size of file
  print 9003,comstring(1:56)
  dstring(1:15)=directory/size
c
  dstring(16:66) = comstring(6:56)
  dstring(16:pathlen-dirlen+15) = path
  status = lib$spawn(dirstring)

c ask user if browse file should be transmitted to terminal
print 9001
read(*,9002)answer
if ((answer .eq. 'n') .or. (answer .eq. 'N')) goto 9999

c print appropriate transfer instructions for modetype
if(modetype .eq. 1) then
  print 9010
  write(20,*) ' Transferring file by KERMIT'
else if (modetype .eq. 2) then
  print 9011
  write(20,*) ' Transferring file by DCL COPY'
else
  print 9012
  write(20,*) ' Transferring file by FTP'
end if

c spawn kermit and send filename from rim pointer
status = lib$spawn(comstring)

c write to audit.log file
  write(20,9020),comstring

c format statements *****
9001 format(' ',/, ' Would you like to transfer this Browse file',
2
9002 format(A1)
9003 format(' ',/,3X,'filename is: ',A56,/)
9010 format(' ',/,3X,'In kermit mode, wait until you get ',
3X,'characters like ,Sp/ @-#1Y',/,
4
3X,'Type alt/k keys together for the VT-KERMIT',
5
3X,'Type receive, and then type connect: ',/)
9011 format(' ',/,3X,'Using the standard VAX DCL COPY command',/,
2
3X,'At the To: type your ',
3
'address/path/file in format: ',/
4
6X,'mode:username password: ',/
5
' device:[path]file.ext',/)
9012 format(' ',/,3X,'At the FTP Awaiting Host> prompt',/,
2
3X,'At the FTP Awaiting Host> prompt',/,
3
3X,'Type SET HOST and your host computer name',/,
4
3X,'Then type LOGIN username and your password',/,
5
3X,'At the FTP> prompt, type SEND/TYPE=IMAGE',/
6
' filename',/,
7
3X,'When prompt returns, type EXIT and continue.',/)
9020 format(' ',4X,A56)
9021 format(A)
9022 format(1)

9999 return

9950 print 9050,ios
9050 format(' ',/,10X,'-ERROR-',12,/,
2
'can not open path.txt file requested in kermispwn')
goto 9999

```

```

end

subroutine lisplat2
*****
c Browse Lisplat2 Subroutine
c -called from directory subroutine
c -prompts user for value for platform from choices
c in 'plat.txt' file for data base query
c
c author: david stoms
c date : 11/06/87
c revised: 02/10/88
c *****
c declarations
c include 'lstoms.src\dir.blk'
c
c integer choice,i,kalen,kclen,MPLATS
c parameter (MPLATS = 10)
c character string*78
c real*8 newrel,oldrel,attrib(1),clause(3)
c get platform type *****
print 9001
10 print 9005
open(unit=7,file='[.text]plat.txt',status='old',
2 err=9910)
c read header of file and MPLATS and continue
read(7,9405) string
read(7,9406) MPLATS
do 90 j=1,MPLATS
  read(7,9407) plat(j)
  print 9006,j,plat(j)
90 continue
close(unit=7)
11 print 9070
call getchoice(choice)
c check for legal values
if ( choice .lt. 1 .or. choice .gt. MPLATS) then
  print 9200,MPLATS
  call pause
  call clrscrn
  goto 10
end if
platform = plat(choice)
c print platform choice to audit.log file
write(20,9020)platform
c if 'all' is chosen, don't project
if (platform .eq. 3HALL) goto 9999
c project from image relation for specified platform
call relselct2(ptr,newrel,oldrel,attrib,kalen)
clause(1) = 8HPLATFORM
clause(2) = 2HEQ
clause(3) = platform
kclen = 3

```



```

print *, '...retrieving data...'
call rimproj(newrel,oldrel,attrib,kalen,clause,kclen)
call prntrow2(newrel)

9999 return

c format statements *****
9001 format(' ',/,30X,'UCSB-BROWSE',/)
9005 format(' ',/, 'Which platform are you interested in?'/,/,
2
9006 format(' ',15X,12,' ',A8)
9020 format(' ',7X,'Platform = ',A8)
9070 format(' ',/, 'Enter choice number: ',)
9200 format(' ',/, 'Choice must be between 1 and ',12)
c format for reading header string in external files
9405 format(A78)
c format for reading MPLATS
9406 format(I)
9407 format(A8)

9910 print 9915,ios
goto 9999
9915 format(' ', ' -ERROR-',12,' ', 'Can not open plat.txt',
2
, ' file requested in listplat subroutine')

end

subroutine lissens2
c*****
c Browse Lissens2 Subroutine
c -called from directory subroutine
c -prompts user for value for sensor from choices
c in 'sensor.txt' file for data base query
c
c author: david stoms
c date : 11/06/87
c revised: 02/10/88
c*****
c declarations
include 'lstoms.src\dir.blk'

integer choice,j,nsens2,kalen,kclen
parameter (NSENS = 10)
real*8 platname,sens(30)
character string*78,tilde*1,fullname*40
real*8 newrel,oldrel,attrib(1),clause(3)

c get sensor type *****
print 9001
print 9091,platform

15 print 9007
2
status='old',err=9910)
c read header of file and continue
read(7,9405) string
16 read(7,9412) tilde

if (tilde .ne. '-') then

```

```

goto 16
end if

c read platform name and number of sensors associated with it
read(7,9410) platname,nsens2
if (platname .ne. platform) goto 16

20 do 90 j=1,nsens2
read(7,9408) sens(j),fullname
print 9008,j,sens(j),fullname
90 continue
close(unit=7)
21 print 9070
call getchoice(choice)

c check for legal values
if ( choice.lt. 1 .or. choice .gt. nsens2) then
print 9200,nsens2
call pause
call clearsreen
goto 15
end if

sensor = sens(choice)
c print sensor choice to audit.log file
write(20,9020)sensor
c if 'all' is chosen, don't project

if (sensor .eq. 3WALL) goto 9999

c project from image relation for specified sensor
call relselct2(ptr,newrel,oldrel,attrib,kalen)
clause(1) = 6HSENSOR
clause(2) = 2HEQ
clause(3) = sensor
kclen = 3
print *, '...retrieving data...'
call rimproj(newrel,oldrel,attrib,kalen,clause,kclen)
call prntrow2(newrel)

9999 return

c format statements *****
9001 format(' ',/,30X,'UCSB-BROWSE',/)
9007 format(' ',/, 'Which sensor type are you interested in?'/,/,
2
9008 format(' ',15X,12,' ',A8,2X,A40)
9020 format(' ',7X,'Sensor = ',A8)
9070 format(' ',/, 'Enter choice number: ',)
9091 format(' ',/, 'Current value of platform = ',A8)
9200 format(' ',/, 'Choice must be between 1 and ',12)
c format for reading header string in external files
9405 format(A78)
c format for reading sens(j),fullname
9408 format(A8,A40)
c format for reading platname and nsens2
9410 format(A8,12)
c format for reading tilde
9412 format(A1)

9910 print 9915,ios
goto 9999
9915 format(' ', ' -ERROR-',12,' ', 'Can not open sensor.txt',
2
, ' file requested in listsens subroutine')

```

```

end
subroutine listcloud
c*****
c Browse Listcloud Subroutine
c -called from catalog subroutines
c -prompts user for value for maximum percent cloud
c cover for data base query
c
c author: david stoms
c date : 10/19/87
c revised: 02/10/88
c*****
c declarations
c include 'lstoms.src\params.blk'
c
c real*8 newrel,oldrel,attrib(1),klaus(3)
c integer kalen,kclen,iklaus(2,3)
c equivalence (klaus(1),iklaus(1,1))
c
c get maximum cloud cover *****
10 print 9000
11 print 9001
12 call getchoice(maxcvr)
13 read(*,9003) maxcvr
14 print maxcvr to audit.log file
15 write(20,9020)maxcvr
c
c check for legal values
16 if ( maxcvr.lt. 0 .or. maxcvr .gt. 100) then
17 print 9200
18 call pause
19 call clearsreen
20 goto 10
21 end if
c
c project from image relation for maximum cloud cover *****
22 call rselct(ptr,newrel,oldrel,attrib,kalen)
23 klaus(1) = 4*HCLD%
24 klaus(2) = 2*HLE
25 iklaus(1,3) = maxcvr
26 kclen = 3
27 print *, '....retrieving data....'
28 call proj(newrel,oldrel,attrib,kalen,klaus,kclen)
9999 return
c format statements *****
9000 format(' ',/,30X,'UCSB-BROWSE',/)
9001 format(' ',/, 'What is the maximum percent cloud cover',
2 ' ',/, 'you want?/')
9002 format(' ',/, 'Enter integer value between 0 and 100 : ',)
9003 format(13)
9020 format(' ',7X,'Maximum Cloud Cover = ',13,'%')
9200 format(' ',/, 'Choice must be between 0 and 100.',/
2 ' ',/, 'Please reenter value.')
```

```

c*****
c Browse Listcont Subroutine
c -called from directory subroutine
c -prompts user for value for continent from choices
c in 'cont.txt' file for data base query
c
c author: david stoms
c date : 11/06/87
c revised: 02/10/88
c*****
c declarations
c include 'lstoms.src\dir.blk'
c
c integer choice,j,NCONTS
c parameter (NCONTS = 11)
c character string*78
c real*8 cont(30,2)
c
c get desired geographic coverage *****
10 print 9000
11 print 9010
12 open(unit=7,file='[.text]cont.txt',status='old',err=9910)
13 c read header of file and NCONTS and continue
14 read(7,9405) string
15 read(7,9406) NCONTS
16 do 90 j=1,NCONTS
17 read(7,9409) cont(j,1),cont(j,2)
18 print 9011,j,cont(j,1),cont(j,2)
19 90 continue
20 close(unit=7)
21 11 print 9070
22 call getchoice(choice)
c
c check for legal values
23 if ( choice.lt.1 .or. choice .gt. NCONTS) then
24 print 9200,NCONTS
25 call pause
26 call clearsreen
27 goto 10
28 end if
29
30 continent(1) = cont(choice,1)
31 continent(2) = cont(choice,2)
32 c print continent to audit.log file
33 write(20,9020)continent(1),continent(2)
c
c if 'all' chosen, directory subroutine will not project a new relation
c in director.for
9999 return
c format statements *****
9000 format(' ',/,30X,'UCSB-BROWSE',/)
9010 format(' ',/, 'Which continent are you interested in?/',
2 ' ',/, 'Choices are: ')
9011 format(' ',5X,12,' ',A8,A8)
9020 format(' ',7X,'Continent = ',2A8)
9070 format(' ',/, 'Enter choice number: ',)
9200 format(' ',/, 'Choice must be between 1 and 12)
c format for reading header string in external files
```

```

9405 format(A78)
c format for reading NCONTS
9406 format(1)
c format for reading cont(j)
9409 format(A8,A8)

9910 print 9915,ios
call pause
goto 9999
9915 format(' ','-ERROR-',12,'Can not open cont.txt',
2
end
subroutine listcount
c*****
c Browse Listcount Subroutine
c -called from directory subroutine
c -prompts user for value for country from choices
c in 'country.txt' file for data base query
c
c author: david stoms
c date : 11/06/87
c revised: 02/10/88
c*****
c declarations
include 'lstoms.src\dir.blk'
integer choice,j,ncount2,kalen,kclen
parameter (NCOUNT = 20)
character string*78,tilde*1,answer*1
real*8 count(30,2),conname(2),newrel,oldrel,attrib(1),
2 clause(17)

```

```

c ask for country in chosen continent *****

```

```

clause(1) = 8HCONTINENT
clause(2) = 2HEQ
clause(3) = 3HALL
clause(4) = 2HOR
clause(5) = 8HCONTINENT
clause(6) = 2HEQ
clause(7) = continent(1)
clause(8) = continent(2)
kclen = 8

```

```

c do you want to look for a specific country?

```

```

10 print 9040,continent(1),continent(2)
read (*,9400)answer

```

```

c if not, then project just based on continent
if ((answer .eq. 'n') .or. (answer .eq. 'N')) goto 500

```

```

c otherwise, list countries for specified continent

```

```

20 print 9001
print 9014
open(unit=7,file='[.txt]country.txt',status='old',
2 err=9910)
read(7,9405) string
26 read(7,9412) tilde

```

```

if (tilde .ne. '-') goto 26
c look for specified conname and number of countries in it
read(7,9413)conname(1),conname(2),ncount2
if (conname(1) .ne. continent(1)) goto 26

27 do 28 j=1,ncount2
read(7,9414) count(j,1),count(j,2)
print 9015,j,count(j,1),count(j,2)
28 continue
close(unit=7)
29 print 9070
call getchoice(choice)

c check for legal values
if (choice .lt. 1 .or. choice .gt. ncount2) then
print 9200,ncount2
call pause
call clearscreen
goto 20
end if

country(1) = count(choice,1)
country(2) = count(choice,2)
c print country choice to audit.log file
write(20,9020)country(1),country(2)
clause(9) = 3HAND
clause(10) = 7HCOUNTRY
clause(11) = 2HEQ
clause(12) = country(1)
clause(13) = country(2)
clause(14) = 2HOR
clause(15) = 7HCOUNTRY
clause(16) = 2HEQ
clause(17) = 3HALL
kclen = 17

c project tuples that match where clause
500 continue
call rselset2(ptr,newrel,oldrel,attrib,kalen)
print *, '...retrieving data...'
call rimprow2(newrel,oldrel,attrib,kalen,clause,kclen)
call prntrow2(newrel)

9999 return

c format statements *****
9001 format(' ',/,30X,'UCSB-BROWSE',/)
9014 format(' ',/, 'Which country are you interested in?///,
2
9015 format(' ',5X,12,' ') ,A8,A8)
9020 format(' ',7X,'Country = ',2A8)
9040 format(' ', 'Do you want to select a specific country in ',
2
AB,A8,'? [Y/N]: ',)
9070 format(' ',/, 'Enter choice number: ',)
9100 format(' ',/, 'Choice must be between 1 and ',12)
9200 format(' ',/, 'Choice must be between 1 and ',12)
9400 format(A)
c format for reading header string in external files
9405 format(A78)
c format for reading tilde
9412 format(A1)
c format for reading conname and ncount2
9413 format(A8,A8,8X,12)
c format for reading count(j,x)

```

```

9414 format(A8,A8)
9910 print 9915,ios
call pause
goto 9999
9915 format(' ','-ERROR-',12/,'Can not open country.txt',
2
,' file requested in listcount subroutine')
end
subroutine listform
c*****
c Browse Listform Subroutine
c -called from directory subroutine
c -prompts user for value for format from choices
c in 'format.txt' file for data base query
c
c author: david stoms
c date : 11/06/87
c revised: 02/10/88
c*****
c declarations
include '[stoms.src]dir.blk'
integer choice,j,kalen,kclen,NFORM
parameter (NFORM = 4)
character string*78
real*8 form(20),newrel,oldrel,attrib(1),clause(3)
c get desired format *****
10 print 9001
print 9018
open(unit=7, file='[.text]format.txt', status='old', err=9910)
c read header of file and NFORM and continue
read(7,9405) string
read(7,9406) NFORM
do 90 j=1,NFORM
read(7,9411) form(j)
print 9019,j,form(j)
90 continue
close(unit=7)
11 print 9070
call getchoice(choice)
c check for legal values
if ( choice.lt. 1 .or. choice .gt. NFORM) then
print 9200,NFORM
call pause
call clearsreen
goto 10
end if
format = form(choice)
c print format choice to audit.log file
write(20,9020),format
kalen = 0
if(format .eq. 3HALL) goto 9999
call rselct2(ptr,newrel,oldrel,attrib,kalen)
clause(1) = 6HFORMAT

```

```

clause(2) = 2HEQ
clause(3) = format
kclen = 3
print *, '...retrieving data...'
call rimprowj(newrel,oldrel,attrib,kalen,kclen)
call pnrrow2(newrel)
9999 return
c format statements *****
9001 format(' ','/30X','UCSB-BROWSE',/)
9018 format(' ','', 'Which format are you interested in?'/,
2
,' Choices are: ')
9019 format(' ','5X,12,') ,A8)
9020 format(' ','7X',Format = ',A8)
9070 format(' ','', 'Enter choice number: ',)
9200 format(' ','', 'Choice must be between 1 and ',12)
c format for reading header string in external files
9405 format(A78)
c format for reading NFORM
9406 format(1)
c format for reading form(j)
9411 format(A8)
9910 print 9915,ios
call pause
goto 9999
9915 format(' ','-ERROR-',12/,'Can not open format.txt',
2
,' file requested in listform subroutine')
end
subroutine listline
c*****
c Browse Listline Subroutine
c -called from catalog subroutine
c -prompts user for value for on-line status
c for data base query
c
c author: david stoms
c date : 10/19/87
c revised: 02/10/88
c*****
c declarations
include '[stoms.src]params.blk'
integer choice,kalen,kclen
real*8 newrel,oldrel,attrib(1),clause(3)
c get on-line status *****
10 print 9001
print 9005
print 9070
call getchoice(choice)
c check for legal values
if ( choice.lt. 1 .or. choice .gt. 3) then
print 9200
call pause
call clearsreen

```

```

      goto 10
    end if
    goto (9999,200,300),choice
  c on-line data only requested
  200 online = 2MON
    goto 500
  c off-line data only requested
  300 online = 3HOFF
  500 continue
  c print on-line choice to audit.log file
    writet(20,9020)online
  call relselct(ptr,newrel,oldrel,attrib,kalen)
  c project from image relation for specified platform
    clause(1) = 6HONLINE
    clause(2) = 2HEQ
    clause(3) = online
    kclen = 3
    print *, '...retrieving data...'
    call proj(newrel,oldrel,attrib,kalen,clause,kclen)
  9999 return

```

```

  c format statements *****

```

```

9001 format(' ',30X,'UCSB-BROWSE',/)
9005 format(' ',5X,'Select On-Line Status: ',/,
2      15X,'1) EITHER -- (on or off-line is ok)',/
3      15X,'2) ON -- (image on-line)',/
4      15X,'3) OFF -- (image off-line)',/)
9020 format(' ',7X,'Online Status = ',A8)
9070 format(' ',/, 'Enter choice number: ',)
9200 format(' ',/, 'Choice must be between 1 and 3')

```

```

end

```

```

subroutine listloc

```

```

c*****
c Browse Listloc Subroutine
c -called from catalog subroutine
c -prompts user for value for coordinates of region,
c specifically for nw lat/long and se lat/long
c -retrieves appropriate records from CATALOG using locproj
c
c author: david stoms
c date : 10/15/87
c revised: 07/25/88
c*****

```

```

c declarations

```

```

  include 'lstoms.src\params.blk'

```

```

  character*74 errmsg
  1 call clearsreen

```

```

  c get location range *****

```

```

  c get northwest latitude
  print 9060

```

```

30 print 9010
print 9050
print 9071
read (*,9001) nwlatd,nwlatm,nwlatz
  c default value chosen
    if (nwlatd .eq. 0) then
      nwlatd = +90
      goto 39
    end if
  c check for legal values for nw latitude
    if (nwlatd .gt. +90) then
      print 9205
      goto 30
    end if
  39 print 9100
  c get northwest longitude
  40 print 9011
print 9051
print 9071
read (*,9002) nwlongd,nwlongm,nwlongz
  c if default chosen for nwlat/long then set selat/long too
  c and skip past prompting for them
    if ((nwlatd .eq. +90) .and.(nwlongd .eq. 0)) then
      nwlongd = -180
      goto 60
    end if
  c check for legal values for nw longitude
    if (nwlongd .lt. -180) then
      print 9206
      goto 40
    end if
  print 9100
  c get southeast latitude
  50 print 9012
print 9050
print 9071
read (*,9001) selatd,selatm,selatz
  c
    if (selatd .eq. 0) then
      selatd = -90
      goto 54
    end if
  c check for legal values for se latitude
    if (selatd .lt. -90) then
      print 9205
      goto 50
    end if
  54 print 9100
  c get southeast longitude
  55 print 9013

```



```

    print 9019,j,nodename(j),fullname
    close(unit=7)
    11 print 9070
    call getchoice(choice)

c check for legal values
if ( choice .lt. 1 .or. choice .gt. NNODE) then
    print 9200,NNODE
    call pause
    call clearsreen
    goto 10
end if

node = nodename(choice)

c print node choice to audit.log file
write(20,9020)node

kalen = 0
if(node .eq. 3HALL) goto 9999
call rselct2(ptr,newrel,oldrel,attrib,kalen)
clause(1) = 8HNODENAME
clause(2) = 2HEQ
clause(3) = node
kclen = 3
print *, '...retrieving data...'
call rimproj(newrel,oldrel,attrib,kalen,clause,kclen)
call prntrow2(newrel)

```

9999 return

c format statements *****

```

9001 format(' ',30X,'UCSB-BROWSE',/)
9018 format(' ',30X,'Which node are you interested in?'/,/,
2
9019 format(' ',5X,12,' ',A8,1X,A60)
9020 format(' ',7X,'Node = ',A8)
9070 format(' ',30X,'Enter choice number: ',/)
9200 format(' ',30X,'Choice must be between 1 and ',12)
c format for reading header string in external files
9405 format(A78)
c format for reading NNODE
9406 format(1)
c format for reading nodename(j),fullname
9411 format(A8,1X,A60)

```

9910 print 9915,ios

call pause

goto 9999

```

9915 format(' ',12,'-ERROR-',12,'Can not open node.txt'
2
', file requested in listnode subroutine')

```

end

subroutine listplat

c*****

```

c Browse Listplat Subroutine
c -called from catalog subroutine
c -prompts user for value for platform from choices
c in 'plat.txt' file for data base query
c
c author: david stoms
c date : 10/19/87

```

```

c revised: 02/10/88
c
c*****
c declarations

include 'lstoms.src\params.blk'

integer choice,j,kalen,kclen,NPLATS
parameter (NPLATS = 10)
character string*78
real*8 plat(30)
real*8 newrel,oldrel,attrib(1),clause(3)

c get platform type *****
print 9001
10 print 9005
open(unit=7,file='(.text\plat.txt',status='old',
2 err=9910)
c read header of file and NPLATS and continue
read(7,9405) string
read(7,9406) NPLATS
do 90 j=1,NPLATS
    read(7,9407) plat(j)
    print 9006,j,plat(j)
90 continue
close(unit=7)
11 print 9070
call getchoice(choice)

c check for legal values
if ( choice .lt. 1 .or. choice .gt. NPLATS) then
    call pause
    call clearsreen
    goto 10
end if

platform = plat(choice)
c print platform choice to audit.log file
write(20,9020)platform

c if 'all' is chosen, don't project
if (platform .eq. 3HALL) goto 9999

c project from image relation for specified platform
call rselct(ptr,newrel,oldrel,attrib,kalen)
clause(1) = 8HPLATFORM
clause(2) = 2HEQ
clause(3) = platform
kclen = 3
print *, '...retrieving data...'
call proj(newrel,oldrel,attrib,kalen,clause,kclen)

9999 return

c format statements *****
9001 format(' ',30X,'UCSB-BROWSE',/)
9005 format(' ',30X,'Which platform are you interested in?'/,/,
2
9006 format(' ',15X,12,' ',A8)
9020 format(' ',7X,'platform = ',A8)

```

```

9070 format(' ',/, ' Enter choice number: ', $)
9200 format(' ',/, ' Choice must be between 1 and ', i2)
c format for reading header string in external files
9405 format(A78)
c format for reading NPLATS
9406 format(i)
c format for reading plat(j)
9407 format(A8)

9910 print 9915, ios
goto 9999
9915 format(' ', ' -ERROR-', i2, ' Can not open plat.txt',
2
end
subroutine listsens
c*****
c Browse Listsens Subroutine
c -called from catalog subroutine
c -prompts user for value for sensor from choices
c in 'sensor.txt' file for data base query
c
c author: david stoms
c date : 10/19/87
c revised: 02/10/88
c*****
c declarations

```

```

include 'stoms.src\params.blk'

integer choice, j, nsens2, kalen, kclen
parameter (NSENS = 10)
real*8 platname, sens(30)
character string*78, tilde*1, fullname*40
real*8 newrel, oldrel, attrib(1), clause(3)

c get sensor type *****
print 9001
print 9091, platform
15 print 9007
open(unit=7, file='[.text]sensor.txt',
2 status='old', err=9910)
c read header of file and continue
read(7, 9405) string
16 read(7, 9412) tilde
if (tilde .ne. '-') then
goto 16
end if

c read platform name and number of sensors associated with it
read(7, 9410) platname, nsens2
if (platname .ne. platform) goto 16

20 do 90 j=1, nsens2
read(7, 9408) sens(j), fullname
print 9008, j, sens(j), fullname
90 continue
close(unit=7)
21 print 9070

```

```

call getchoice(choice)
c check for legal values
if ( choice .lt. 1 .or. choice .gt. nsens2) then
print 9200, nsens2
call pause
call clearscreen
goto 15
end if

sensor = sens(choice)
c print sensor choice in audit.log file
write(20, 9020) sensor

c if 'all' is chosen, don't project
if (sensor .eq. 3HALL) goto 9999

c project from image relation for specified sensor
call rselc(ptr, newrel, oldrel, attrib, kalen)
clause(1) = 6HSENSOR
clause(2) = 2HEQ
clause(3) = sensor
kclen = 3
print *, ' ---retrieving data---'
call proj(newrel, oldrel, attrib, kalen, clause, kclen)

9999 return

c format statements *****
9001 format(' ', /30X, 'UCSB-BROWSE', /)
9007 format(' ', /, ' which sensor are you interested in? ', /,
2
9008 format(' ', 15X, i2, ' ', A8, 2X, A40)
9020 format(' ', 7X, 'Sensor = ', A8)
9070 format(' ', /, ' Enter choice number: ', $)
9091 format(' ', /, ' Current value of platform = ', A8)
9200 format(' ', /, ' Choice must be between 1 and ', i2)
c format for reading header string in external files
9405 format(A78)
c format for reading sens(j), fullname
9408 format(A8, A40)
c format for reading platname and nsens2
9410 format(A8, i2)
c format for reading tilde
9412 format(A1)

9910 print 9915, ios
goto 9999
9915 format(' ', ' -ERROR-', i2, ' Can not open sensor.txt',
2
end
subroutine listtime(menuid)
c*****
c Browse Listtime Subroutine
c -called from catalog subroutine
c -prompts user for value for start date and end date
c of period to search on
c

```



```

c  author: david stoms
c  date : 12/01/87
c  revised: 02/10/88
c *****

```

```

c declarations

```

```

include 'stoms.src'params.blk'

```

```

integer choice
character menuid*10,answer*1
parameter (NATTS =6)
real*8 newrel,oldrel,attrib(1),klaus(20)
integer kalen,kclen,iklaus(2,20)
equivalence (klaus(1),iklaus(1,1))

c get time specs *****

```

```

1 call menu(menuid)
call getchoice(choice)

```

```

c check for legal values
if(choice .gt. NATTS) then
print 9200,NATTS
call pause
goto 1
end if

```

```

call clearsreen

```

```

print 9001
goto (100,200,300,400,500,9900),choice

```

```

c get start date *****

```

```

100 print 9014
print 9051
print 9071
read (*,9304) strtjr,strtmon

```

```

c check for legal values
if ((strtmon .gt. 12) .or. (strtmon .lt. 1)) then
print 9207
goto 100
end if

```

```

c call proj *****

```

```

klaus(1) = 4HYEAR
klaus(2) = 2HEQ
iklaus(1,3) = strtjr
klaus(4) = 3HAND
klaus(5) = 4HMNTH
klaus(6) = 2HGE
iklaus(1,7) = strtmon
klaus(8) = 2HOR
klaus(9) = 4HYEAR
klaus(10) = 2HGT
iklaus(1,11) = strtjr
kclen = 11

```

```

print *, '...retrieving data...'
call proj(newrel,oldrel,attrib,kalen,klaus,kclen)

```

```

c get end date *****

```

```

150 print 9015
print 9051
print 9071

```

```

read (*,9304) endyr,endmon

```

```

c check for legal values
if ((endmon .gt. 12) .or. (endmon .lt. 1)) then
print 9207
goto 200
end if

```

```

c print year & month to audit.log file
write(20,9020)strtjr,strtmon,endyr,endmon

```

```

c call proj for end date*****
c only change some of the klaus values
call rselct(ptr,newrel,oldrel,attrib,kalen)

```

```

c klaus(1) = 4HYEAR
c klaus(2) = 2HEQ
c iklaus(1,3) = endyr
c klaus(4) = 3HAND
c klaus(5) = 4HMNTH
c klaus(6) = 2HLE
c iklaus(1,7) = endmon
c klaus(8) = 2HOR
c klaus(9) = 4HYEAR
c klaus(10) = 2HLT
c iklaus(1,11) = endyr
kclen = 11

```

```

print *, '...retrieving data...'
call proj(newrel,oldrel,attrib,kalen,klaus,kclen)

```

```

goto 1

```

```

c still need to check that bdate < edate
c call checkdate---to be written

```

```

c get single date *****

```

```

200 print 9017
print 9052
print 9071

```

```

read (*,9300) strtjr,strtmon,strtday
c check legal values
if((strtmon .gt. 12) .or. (strtmon .lt. 1) .or.
2 (strtday .gt. 31)) then
print 9208
goto 300
end if

```

```

c print date to audit.log file

```

```

write(20,9021)strtjr,strtmon,strtday

```

```

call rselct(ptr,newrel,oldrel,attrib,kalen)
klaus(1) = 4HMNTH
klaus(2) = 2HEQ
iklaus(1,3) = strtmon
klaus(4) = 3HAND
klaus(5) = 4HYEAR
klaus(6) = 2HEQ
iklaus(1,7) = strtjr
klaus(8) = 3HAND
klaus(9) = 3HDAY
klaus(10) = 2HEQ
iklaus(1,11) = strtday
kclen = 11

```

```

print *, '...retrieving data...'
call proj(newrel,oldrel,attrib,kalen,klaus,kclen)
goto 1

```

```

c get single month *****
300 print 9016
read (*,9301) strtmon
c check legal values
if(strtmon.gt. 12) then
  print 9209
  goto 300
end if

c print month to audit.log file
write(20,9022)strtmon

call rselcct(ptr,newrel,oldrel,attrib,kalen)
klausel(1) = 4*MMNH
klausel(2) = 2HEQ
iklausel(1,3) = strtmon
kclen = 3
print *, '....retrieving data....'
call proj(newrel,oldrel,attrib,kalen,klausel,kclen)
goto 1

c display time specs *****
400 print *, 'time specs will go here'
call pause
goto 1

c clear/restart time specs

500 print 9050
read (*,9302) answer
if((answer(1:1).eq. 'n').or. (answer.eq. 'N')) then
  goto 1
end if

c otherwise, set values to zero
strtrm = 01
strtrmon = 01
strtday = 01
endyr = 99
endmon = 12
endday = 31
call rimrem(newrel)
goto 1

c format statements *****
9001 format(' ',30X,'UCSB-BROWSE',/)
9014 format(' ',/) Enter start date in format YYMM'//)
9015 format(' ',/) Enter end date in format YYMM'//)
9016 format(' ',/) Enter desired month in format MM'//)
9017 format(' ',/) Enter date in format YYMMDD'//)
9020 format(' ',7X,'Start date = ',12.2,' ',12.2,6X,'End date = ',
2 12.2,' ',12.2)
9021 format(' ',7X,'Date = ',12.2,' ',12.2,' ',12.2)
9022 format(' ',7X,'Month = ',12.2)
9050 format(' ',/) Are you sure? [Y/N]: ',$)
9051 format(' ',/) where YY = year, MM = month'//)
9052 format(' ',/) where YY = year, MM = month, DD = day'//)
c 9070 format(' ',/) Enter choice: ',$)
9071 format(' ',/) Enter value: ',$)
9100 format(' ',/) -----'//)
9200 format(' ',/) Choice must be between 1 and '12,/'//)
9207 format(' ',/) Values must be: 1<=MM<=12 ' ,/

```

```

2 9208 format(' ',/,) Please reenter values.'//)
2 Values must be: 1<=MM<=12; '
3 DD<=31.'//)
3 Please reenter values.'//)
9209 format(' ',/,) Values must be: 1<=MM<=12 ' )
9300 format(12,12,12)
9301 format(12)
9302 format(a)
9304 format(12,12)

9900 return
end

subroutine locproj
*****
c Browse Locproj Subroutine
c -called from mapwd and listloc subroutines
c -projects tuples from locator relation and intersects
c with the current temporary relation
c
c author: david stoms
c date : 11/23/87
c revised: 07/28/88
c *****
c declarations
include '[stoms.src]params.blk'

real*8 newrel,oldrel,atts(3),klausel(15),attrib(14),
2 firstold,oldatt,firstatt,relopr
integer kalen,kclen,kalen2,kclen2
character ermsg*74
integer numrows

c print lat/long values to audit.log file
write(20,9020)nwlat,nwlong,selat,selong
c write(*,9020)nwlat,nwlong,selat,selong
c call pause

c project from locator relation limited by lat/long specs *****
call rselcct(ptr,newrel,oldrel,atts,kalen)
c reset values for oldrel, atts, and kalen for proj from locator
firstold = oldrel
oldrel = 7HLOCATOR
atts(1) = 8HSCENENUM
kalen = 1
klausel(1) = 8HCENT_LAT
klausel(2) = 2HLE
klausel(3) = nwlat
klausel(4) = 3HAND
klausel(5) = 8HCENT_LAT
klausel(6) = 2HGE
klausel(7) = selat
klausel(8) = 3HAND
klausel(9) = 8HCENT_LOW
klausel(10) = 2HGE
klausel(11) = nwlong
klausel(12) = 3HAND
klausel(13) = 8HCENT_LOW
klausel(14) = 2HLE
klausel(15) = selong
kclen = 15

```

```

call clearsreen
print *, '...retrieving data....'
call rimproj(newrel, oldrel, atts, kalen, klaus, kclen)
call zerohits(newrel, numrows)
call printrow(newrel)
c if no records retrieved, do not continue join/project
if (numrows .eq. 0) goto 9999
c now join this temp relation with current answer relation
call relselct(ptr, newrel, oldrel, attrib, kalen)
oldatt = 8HSCENENUM
firstatt = 7HSCENEID
relopr = 2HEQ
c call rimjoin(oldrel, oldatt, firstold, firstatt, newrel, relopr)
call rimjoin(firstold, firstatt, oldrel, oldatt, newrel, relopr)
call zerohits(newrel, numrows)
c call printrow(newrel)
c project the image attributes only from temp relation
call relselct(ptr, newrel, oldrel, attrib, kalen)
kalen2 = 14
attrib(1) = 7HSCENEID
attrib(2) = 4HCLDX
attrib(3) = 4HYEAR
attrib(4) = 4HMNTH
attrib(5) = 3HDAY
attrib(6) = 8HPHYDESC
attrib(7) = 7HQUALITY
attrib(8) = 7HHISTORY
attrib(9) = 6HONLINE
attrib(10) = 6HFORMAT
attrib(11) = 6HSENSOR
attrib(12) = 8HPLATFORM
attrib(13) = 4HMISS
attrib(14) = 7HPOINTER
kclen2 = 0
klaus(1) = 1H0
call rimproj(newrel, oldrel, attrib, kalen2, klaus, kclen2)
call printrow(newrel)

c format statements *****
9020 format(' ', 6X, 'NW Lat =', SP, F11.6, 6X, 'NW Long =', F11.6, //,
2       7X, 'SE Lat =', F11.6, 6X, 'SE Long =', F11.6)
9999 return
end

c*****
c Browse Main Program
c displays welcome banner, calls mainmenu
c
c author: david stoms
c date : 12/07/87
c revised : 07/07/88
c*****
c declarations
common /vers/version, revdate
common /term/termtype
common /userdata/username, institution, discipline,
2 hardware, access
2 character revdate*8, version*3, username*40, institution*60,
discipline*40, hardware*30, access*20
integer ios, termtype
character menuid*10, today*9, now*8, filename*18

```

```

c open an audit file to track system use
open(unit = 20, file = 'audit.log', access = 'append',
2 status = 'unknown', err = 9910)
print 9001

c get system time and date
call date(today)
call time(now)

print 9005, today, now
write(20, 9006)
write(20, 9005) today, now

c read version and revdate from version.txt file
filename = '[.text]version.txt'
open(unit = 7, file = filename, status = 'old', err = 9007)
c read header and continue
read (7, 9011) string
read (7, 9013) version
read (7, 9014) revdate
close(unit=7)

c find out what terminal type for clearsreen codes
20 call setterm

c if tek 4010 terminal, use graphic banner
if (termtype .eq. 2) then
call bantek
goto 50
end if

c else print banner in ascii mode
menuid = 'menu00.txt'
call banner(menuid)

50 call pause

c get userdata
call userdata

c open another file in [stoms] directory to track system use
open(unit = 22, file = '$disk2:[stoms]browse.log',
2 access = 'append', status = 'unknown', err = 9920)

write(22, 9016) username, today, now
9016 format(' ', 3X, A40, 5X, A6, 5X, A8)
close(unit=22)

1 menuid = 'menu01.txt'
call mainmenu(menuid)
goto 1

c format statements *****
9001 format(' ', //, 20X, 'UCSB-RSRU BROWSE TESTBED', //)
9005 format(' ', 6X, 'Date: ', A9, 6X, 'Current Pacific Coast Time: ',
2 A8, //)
.9006 format(' ', //,
2 ' ', //,
3 7X, 'Start of BROWSE job.', //)
c 9006 format(' ', //, 7X, 'Start of BROWSE job.', //)

9011 format(A78)
9013 format(A3)
9014 format(A8)

```

```

c Error routines *****
c
9910 print 9925
9925 format(' ', ' ...BROWSE program already in use.', /,
2         ' Please try again later...')
    goto 100
c
9907 print 9008, ios, filename
9908 format(' ', ' -ERROR- ', 12,
2         ' : Can not open ', A11, ' file.')
    call pause
    goto 20
c
9920 print 9017, ios
9917 format(' ', ' ,', ' -ERROR-', can not open BROWSE.LOG file')
    call pause
    goto 1
c
100 end
    subroutine mainmenu(menuid)
c*****
c Browse Main Menu Subroutine
c called by main program and other submenus with menu01.txt
c calls menu.sub for main menu, reads choice
c
c author: david stoms
c date : 12/01/87
c revised: 02/09/88
c*****
c declarations
integer choice, numlines
character menuid*10, string*72
c print menu to terminal
1 call menu(menuid)
call getchoice(choice)
c check legal values
if (choice .lt. 1 .or. choice .gt. 5) then
    print 9000
    call pause
    goto 1
endif
goto(100, 200, 300, 400, 500), choice
c query data base choice
100 menuid = 'menu11.txt'
call query(menuid)
goto 9999
c goto image browsing choice
200 call clearscreen
print 9003
open(unit=7, file='1.txt', status='old',
2 err=9900)
c read header of ids.txt file and ignore it
read(7, 9001) string
read(7, 9002) numlines
print 9003
do 220 i=1, numlines
    read(7, 9001) string

```

```

220 continue
    print 9004, string
    call pause
    close(unit=7)
    goto 9999
c help choice
300 call help
    goto 9999
c mail choice
400 call messages
    goto 9999
c quit choice
500 call quit
c will eventually do housekeeping and update user database
c format statements *****
9000 format(' ', /, ' Choice must be between 1 and 5')
9001 format(A)
9002 format(I)
9003 format(' ', /, 12X, 'UCSB-BROWSE', /)
9004 format(' ', 5X, A)
9900 print 9910, ios
9910 format(' ', ' -ERROR-', 12, /, 'Cannot open ids.txt file',
2         ' requested in mainmenu.')
9999 return
end
c ***** SUBROUTINE FOR GKS CHOICE FUNCTION--STOMS 7/22/88
c BASED ON EXAMPLE ON PAGE 6-63 IN GKS REF MANUAL
c ALLOWS USER TO POINT WITH CURSER AT MENU CHOICE
SUBROUTINE MAPCHOICE(INPUT_CHOICE)
IMPLICIT NONE
INTEGER DISP, DATA_RECORD(3), NUM_CHOICES, SIZES(4),
2 ADDRESSES(4), PROMPT_ECHO_TYPE, ERROR_STATUS,
3 INPUT_MODE, ECHO_FLAG, RECORD_BUFFER_LENGTH, RECORD_SIZE,
4 INPUT_STATUS, INITIAL_CHOICE, DEVICE_NUM, INPUT_CHOICE,
5 INITIAL_STATUS
REAL ECHO_AREA(4)
CHARACTER*80 CURRENT_STRINGS(4)
DATA DISP /1/, DEVICE_NUM /1/
c FIRST ELEMENT IN DATA RECORD IS NUMBER OF CHOICES
EQUIVALENCE(DATA_RECORD(1), NUM_CHOICES)
c ESTABLISH SIZE OF RECORD BUFFER: 12 BYTES
RECORD_BUFFER_LENGTH = 12
DATA_RECORD(2) = %LOC(SIZES(1))
DATA_RECORD(3) = %LOC(ADDRESSES(1))
ADDRESSES(1) = %LOC(CURRENT_STRINGS(1))
ADDRESSES(2) = %LOC(CURRENT_STRINGS(2))
ADDRESSES(3) = %LOC(CURRENT_STRINGS(3))
ADDRESSES(4) = %LOC(CURRENT_STRINGS(4))

```

```

NUM_CHOICES = 4
C SEE PAGE 10-120 OF REF MANUAL
CALL GKSSNO CHOICE STATE(DISP, DEVICE_NUM,
2 ERROR STATUS, INPUT_MODE, ECHO_FLAG, INITIAL_STATUS,
3 INITIAL_CHOICE, PROMPT ECHO TYPE, ECHO_AREA,
4 DATA_RECORD, RECORD_BUFFER_LENGTH, RECORD_SIZE)

```

```

PROMPT ECHO TYPE = 1
INITIAL_CHOICE = 2
ECHO_AREA(1) = 300.0
ECHO_AREA(2) = 1200.0
ECHO_AREA(3) = 2350.0
ECHO_AREA(4) = 2800.0

```

```

C ESTABLISH SIZE OF PROMPT STRINGS
SIZES(1) = 24
SIZES(2) = 24
SIZES(3) = 24
SIZES(4) = 24

```

```

C SET TEXT HEIGHT
C THIS DOESN'T WORK FOR CHOICES
CALL GSTXPR(2,2)
CALL GSCHH(0,15)

```

```

C ESTABLISH LOCATIONS OF PROMPT STRINGS
ADDRESSES(1) = %LOC( '1: ZOOM ' )
ADDRESSES(2) = %LOC( '2: UNZOOM ' )
ADDRESSES(3) = %LOC( '3: GO BACK TO LAST ZOOM ' )
ADDRESSES(4) = %LOC( '4: SEARCH IN THIS WINDOW' )

```

```

CALL GKSSINIT CHOICE(DISP, DEVICE_NUM,
2 INITIAL_STATUS, INITIAL_CHOICE, PROMPT ECHO TYPE,
3 ECHO_AREA, DATA_RECORD, RECORD_BUFFER_LENGTH)

```

```

9000 2 FORMAT( ' ',//, ' XMIN = ',F8.2,2X, 'XMAX = ',F8.2,2X,
/, ' YMIN = ',F8.2,2X, 'YMAX = ',F8.2,2X)

```

```

C CALL GKSSSET_CHOICE MODE(DISP, DEVICE_NUM,0,1)
2 GKSSK_INPUT_MODE_REQUEST, GKSSK_ECHO)

```

```

CALL GKSSREQUEST CHOICE(DISP, DEVICE_NUM, INPUT_STATUS,
2 INPUT_CHOICE)

```

```

C OUTPUT CHOICE NUMBER
C WRITE(6,*) INPUT_CHOICE

```

```

RETURN
END

```

```

SUBROUTINE MAPUD IBROWSE MODIFICATION

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C UCSB BROWSE: WORLD MAP ZOOM PROGRAM C
C
C MARK FRIEDL AND KEN MCGWIRE C
C
C REMOTE SENSING RESEARCH UNIT C
C UNIVERSITY OF CALIFORNIA SANTA BARBARA C
C
C JAN 29, 1988 C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

C HOST MACHINE: MicroVAX II
C LANGUAGE: DEC FORTRAN
C GRAPHICS TOOLS: GK$ - LEVEL 0b
C EDITED FOR BROWSE--AUGUST 29, 1988 BY STOMS

```

```

C THIS PROGRAM WILL DISPLAY CONTINENTS AND ALLOW THE USER
C TO ZOOM TO SUCCESSIVELY FINER LEVELS OF DETAIL USING THE
C MOUSE TO SPECIFY ZOOM COORDINATES. THE PROGRAM IS STRUC-
C TURED INTO TWO MAIN LEVELS. AT THE TOP LEVEL, THE USER
C IS PRESENTED WITH A MENU AND ASKED TO SELECT A CONTINENT.
C THE COORDINATE DATA FOR THE SELECTED CONTINENT IS THEN
C READ IN. THE GIVEN CONTINENT IS PLOTTED, AND THE USER IS
C PRESENTED WITH A SECONDARY LEVEL MENU. THIS MENU ALLOWS
C THE USER TO ZOOM, UNZOOM, OR RETURN TO THE TOP LEVEL MENU
C WHERE HE OR SHE CAN EITHER SELECT A NEW CONTINENT, OR QUIT
C THE PROGRAM.

```

```

C THE VECTOR DATABASE IS STRUCTURED HIERARCHICALLY IN ORDER TO
C BOTH MINIMIZE ACCESS TIME AND PLOTTING TIME, AS WELL AS TO
C LOGICALLY SEGMENT THE DATA FILES ACCORDING TO GEOGRAPHIC AREA.
C IN THIS MANNER A SET OF DATA FILES IS ASSOCIATED WITH EACH CONTINENT.
C AT THE HIGHEST LEVEL IS A VECTOR FILE CONTAINING A CRUDE OUTLINE
C OF THE CONTINENT INCLUDING GROSS POLITICAL AND PHYSIOGRAPHIC
C FEATURES. BELOW THIS ARE TWO LEVELS OF VECTOR FILES WHICH
C PORTRAY SUCCESSIVELY FINER DEGREES OF RESOLUTION. WHEN A CONTINENT
C OF INTEREST IS SELECTED BY THE USER, THREE DATA FILES ARE THEN
C SELECTED WHICH ARE USED UNTIL A NEW CONTINENT IS SELECTED OR
C THE PROGRAM IS EXITED. 1) COARSE RESOLUTION VECTOR FILE OF CONTINENT
C OUTLINE WHICH SERVES AS INITIAL MAP AND REFERENCE MAP ONCE USER
C ZOOMS INTO REGION OF INTEREST. 2) LOOKUP TABLE FILE OF HIGHER
C RESOLUTION VECTOR FILES AND THEIR GEOGRAPHIC WINDOW. 3) POINT FILE OF
C POINT LOCATIONS IN THE CORRESPONDING CONTINENT.

```

```

C THE LOOKUP TABLE FILE LISTS THE VARIOUS VECTOR DATA FILES WHICH ARE
C ASSOCIATED WITH EACH CONTINENT. ASSOCIATED WITH EACH ENTRY IS A
C FILE NAME, GEOGRAPHIC WINDOW, AND A LEVEL NUMBER. THE LEVEL NUMBER
C INDICATES WHAT LEVEL OF RESOLUTION THE SPECIFIC FILE REPRESENTS.
C WHEN A SPECIFIC ZOOM WINDOW IS SELECTED BY THE USER, THE ZOOM LEVEL
C CALCULATED BASED UPON THE LATITUDE AND LONGITUDE RANGES, AND THE
C LOOKUP TABLE IS SEARCHED FOR VECTOR FILES WHICH ARE CONTAINED BY,
C OR OVERLAY THE ZOOM WINDOW, AND WHICH ARE AT THE APPROPRIATE LEVEL
C OF RESOLUTION

```

```

C ***** VARIABLE LISTINGS *****

```

```

C DISP,WORKST: WORKSTATION PARAMETERS FOR GK$

```

```

C SYSERROR: SYSTEM ERROR MESSAGE FILE

```

```

C ASF: LIST OF ASPECT SOURCE FLAGS FOR GK$

```

```

C STAT: STATUS FLAG FOR MOUSE INPUT

```

```

C TNR: TRANSFORMATION NUMBER

```

```

C FLAG: FLAG INDICATING ZOOMED STATUS

```

```

C CHOICE: INPUT CHOICE FOR CONTINENT

```

```

C CURTOT: ARRAY USED TO COUNT NUMBER OF PTS IN EACH VECTOR

```

```

C LEVELNO: INTEGER INDICATING CURRENT RESOLUTION OF ZOOMING

```

```

C DIGTBL: FILENAME OF CURRENT LOOKUP TABLE

```

```

C CONT: CURRENT CONTINENT SELECTED

```

```

C REFER: FILENAME OF CURRENT REFERENCE VECTOR FILE

```

```

C PTFIL: FILENAME OF CURRENT POINT FILE

```

```

C X1,X2,Y1,Y2: COORDINATES OF CURRENT ZOOM WINDOW

```

```

C OX1,OX2,OY1,OY2: COORDINATES USED TO PLOT REFERENCE BOX ON REFERENCE MAP
C XMN,XMX,YMN,YMX: COORDINATES OF WINDOW FOR CURRENT CONTINENT

```

```

C ZOOMED: LOGICAL INDICATING WHETHER MAP IS CURRENTLY ZOOMED

```

```

C CURLEVEL: COUNTS NUMBER OF TIMES ZOOMED ON MAP

```

```

C XREC,YREC: VECTORS TO RECORD COORDS OF EACH ZOOM

```

```

C   FORTRAN BINDING FOR GKS:
C   INCLUDE 'SYS$LIBRARY:GKSDEF.S.BND'
C   INCLUDE 'ISTONS.SRC\PARAMS.BLK'
C   ***** DECLARATIONS *****
C   ***** BROWSE MODIFICATION *****

INTEGER      DISP,SYSERROR,TYPE,I,ASF(13),WORKST
INTEGER      STAT,TNR,CHOICE,CURTOT(2000)
INTEGER      LEVELNO,CURLEVEL
CHARACTER*18 PROMPT,DIGTBL,CONT,REFER,PTFIL
REAL         X1,X2,Y1,Y2,CURX(10000),CURY(10000)
REAL         XMN,XXM,YMN,YMX,CXMN,CXXM,CYMN,CYMX
REAL         OX1,OX2,OY2,OY1,XREC(10),YREC(10)
LOGICAL      ZOOMED,FLAG

C*****  CONSTANTS  *****
DATA DISP/1/,SYSERROR/0/
DATA TNR1/1/,TNR2/2/,TNR3/3/,TNR4/4/

C*****  BEGIN CODE *****
FLAG = .FALSE.

C ** 41 = NATIVE - 72 = TEK 4014 **
WORKST = 72

C NO NEED TO OPEN GKS WORKSTATION IN MAPUD FOR BROWSE
C ** SET UP WORKSTATION **
C CALL GOPKS(SYSERROR)
C CALL GOPK(DISP,GKSK_CONID_DEFAULT,WORKST)
C CALL GACK(DISP)

C CALL GSDS(DISP,0,0)

C ** SET UP BUNDLE ATTRIBUTES **
DO 27 I=1,13
  ASF(I)=1
27

CALL GSASF(ASF)

C *   CHOOSE WHICH CONTINENT AND ESTABLISH FILE *
C *   NAMES AND LAT/LON BOUNDARIES *

C BROWSE MODIFICATION IN SUBROUTINE NAME
28 CALL MAPMENU(REFER,CXMN,CXXM,CYMN,CYMX,CONT,PTFIL,DIGTBL)
IF (REFER.EQ.'QUIT') GOTO 600
XMN=CXMN
XXM=CXXM
YMN=CYMN
YMX=CYMX

CURLEVEL = 0
LEVELNO = 4

CALL SETWIN(XMN,XXM,YMN,YMX,OX1,OX2,OY1,OY2)

C *   READ CONTINENT DATA FILES INTO ARRAY *
26 CALL READCURRENT(CURX,CURY,REFER,CURTOT)

CALL GCLRUK(1,1)
CALL SETUP(CXMN,CXXM,CYMN,CYMX,XMN,XXM,YMN,YMX,FLAG)

C *   PLOT REFERENCE IMAGE *

```

```

IF (LEVELNO.NE.4) THEN
CALL GSELNT(2)
CALL PLOTPTS(CURX,CURY,CURTOT)
CALL DRAWBOX(OX1,OX2,OY1,OY2)
ENDIF

C *   PLOT MAIN IMAGE *
CALL GSELNT(5)
ZOOMED = .FALSE.

29 CALL SETLEVEL(LEVELNO,OX1,OX2,OY1,OY2)

C BROWSE MODIFICATION
IF (REFER.EQ.'[.MAPS]WORLD3.OUT'.AND.LEVELNO.EQ.1) THEN
RUE.) CALL READCURRENT(CURX,CURY,REFER,CURTOT)
CALL PLOTPTS(CURX,CURY,CURTOT)
ELSE
CALL PLOT DIGFILS(DIGTBL,LEVELNO,XMN,XXM,YMN,YMX)
IF (LEVELNO.EQ.3) THEN
DIGTBL='[.MAPS]PHYS.TBL' IBROWSE MODIFICATION
CALL GSELNT(2)
CALL PLOT DIGFILS(DIGTBL,LEVELNO,XMN,XXM,
+ YMN,YMN)
DIGTBL='[.MAPS]TILE.TBL'
CALL GSELNT(1)
ENDIF
ENDIF

C *   PLOT POINTS *
CALL READPTS(PTFIL,XMN,XXM,YMN,YMX,LEVELNO)

C *   ANNOTATE DISPLAY *
CALL GSELNT(3)
CONT = REFER
CALL TEXT(XMN,XXM,YMN,YMX,CXMN,CXXM,CYMN,CYMX,CONT,LEVELNO)

C *   CHOOSE ZOOM, UNZOOM, MAIN MENU *
C *   ALLOWS FOR GKS CHOOSE FUNCTION WITH MOUSE POINTING *
CALL MAPCHOICE(CHOICE) IBROWSE MODIFICATION

IF(CHOICE.EQ.1) THEN
CALL GTX(0.05,0.6,'>')
FLAG = .TRUE.
CALL GSELNT(5)
CALL CHOOSEWIN(X1,X2,Y1,Y2,STAT,1,DISP,XMN,XXM,YMN,YMX)
CALL SETWIN(X1,X2,Y1,Y2,OX1,OX2,OY1,OY2)
CALL ZOOM(X1,X2,Y1,Y2,XMN,XXM,YMN,YMX)
CALL SETBOUND(X1,X2,Y1,Y2,XMN,XXM,YMN,YMX)
CALL RECORD_WINDOW(XMN,XXM,YMN,YMX,XREC,YREC,CURLEVEL)
CALL PICK CONT(XMN,XXM,YMN,YMX,CXMN,CXXM,CYMN,CYMX,REFER,PTFIL)
CALL GSNM(2,CXMN,CXXM,CYMN,CYMX)
CALL GSVP(2,0.7,1.0,0.7,1.0)
ZOOMED = .TRUE.
GOTO 26

ELSE IF(CHOICE.EQ.2) THEN
CALL GTX(0.05,0.475,'>')
CALL UNZOOM(XMN,XXM,YMN,YMX,CXMN,CXXM,CYMN,CYMX)
CALL SETWIN(XMN,XXM,YMN,YMX,OX1,OX2,OY1,OY2)

```

```

FLAG = .FALSE.
ZOOCHED = .FALSE.
LEVELNO = 1
CURLEVEL = 0
GOTO 28
ELSE IF(CHOICE .EQ. 3) THEN
  CALL GTX(0.05,0.33,'>')
  CALL GOBACK(XNM, XMX, YNM, XMX, XREC, YREC, CURLEVEL,
+           CXNM, CMX, CYNM, CYMX)
  CALL SETWIN(XNM, XMX, YNM, XMX, OX1, OX2, OY1, OY2)
  FLAG = .FALSE.
  CALL PICK_CONT(XNM, XMX, YNM, XMX, CMXN, CMX, CYNM, CYMX, REFER, PTFIL)
  GOTO 26
ELSE
  CALL GTX(0.05,0.20,'>')
  FLAG = .FALSE.
  GOTO 600
ENDIF

C SET LAT/LONG VALUES FOR RIM QUERY IBROWSE MODIFICATIONS
600 CONTINUE
  NWLAT = YMX
  NWLONG = XNM
  SELAT = YNM
  SELONG = XMX
  C PROJECT TUPLES FROM LOCATOR RELATION IN INVENTORY DB
  CALL LOC PROJ
  C SEL TEXT HEIGHT BACK TO DEFAULT
  CALL GK$SET TEXT HEIGHT(0.01)
  C DON'T NEED TO CLOSE WORKSTATION IN MAPMD FOR BROWSE
  C CALL GDANK(DISP)
  C CALL GCLWK(DISP)
  C CALL GCLKS()
10  FORMAT(2F8.3)
13  FORMAT(11)

C STOP
  RETURN
  END

C IBROWSE MODIFICATION

C ***** END OF MAIN MAPMD PROGRAM *****
C !!!!!!!!!!!!!!! SUBROUTINE DEFINITIONS !!!!!!!!!!!!!!!
C ***** SET REFERENCE MAP FILE *****
  SUBROUTINE PICK_CONT(XNM, XMX, YNM, XMX, CMXN, CMX, CYNM, CYMX, REFER, PTFIL)
    REAL XNM, XMX, YNM, XMX, CMXN, CMX, CYNM, CYMX, CX1, CX2, CY1, CY2
    REAL PX, PY
    CHARACTER*18 REFER, FILNAM, PTFIL
    INTEGER X1, X2, Y1, Y2, ERROR
    LOGICAL INSIDE
    REFER = 'YAHOO'

    OPEN(UNIT=7, FILE='[.MAPS]FILE.TBL', IOSTAT=ERROR, STATUS='OLD',
+      FORM = 'FORMATTED', READONLY)
    IBROWSE MODIFICATION

```

```

20 READ(7,21,END=22) FILNAM,X1,X2,Y1,Y2,LNO,PTFIL

  CX1 = REAL(X1)
  CX2 = REAL(X2)
  CY1 = REAL(Y1)
  CY2 = REAL(Y2)
  INSIDE = .FALSE.
  IF (XNM .GT. CX1 .AND. XMX .LT. CX2 .AND.
+    YNM .GT. CY1 .AND. YMX .LT. CY2) THEN
    REFER = FILNAM
  ELSE
    GOTO 20
  ENDIF
  CLOSE(UNIT=7)

21 FORMAT(A18,2X,14,1X,14,1X,13,1X,12,1X,11,1X,A18) IBROWSE MODIFICATION

22 IF (REFER .EQ. 'YAHOO') THEN
  REFER = '[.MAPS]WORLD3.OUT'
  PTFIL = '[.MAPS]WORLD.PTS'
  ELSE
    CXNM = CX1
    CXMX = CX2
    CYNM = CY1
    CYMX = CY2
  ENDIF
  RETURN
  END

C ***** SET RESOLUTION LEVEL FOR DIG FILES *****
  SUBROUTINE SETLEVEL(LEVELNO,OX1,OX2,OY1,OY2)

  INTEGER LEVELNO
  REAL OX1,OX2,OY1,OY2
  REAL X RANGE, Y RANGE

  X RANGE = ABS(OX1-OX2)
  Y RANGE = ABS(OY1-OY2)

  IF (X RANGE .LT. 10 .OR. Y RANGE .LT. THAN 10) THEN
    LEVELNO = 3
  ELSE IF (X RANGE .LT. 50 .OR. Y RANGE .LT. 50) THEN
    LEVELNO = 2
  ELSE
    LEVELNO = 1
  ENDIF
  RETURN
  END

C ***** SEARCH LOOKUP TABLE AND PLOT APPROPRIATE FILES *****
  SUBROUTINE PLOT_DIGFILS(DIGTBL,LEVELNO,XNM,XMX,YNM,YMX)
  CHARACTER*18 DIGTBL,FILNAM,PTFIL,DIGFIL
  INTEGER LEVELNO,ERROR,LNO,UPPER,CURTOT(2000)
  INTEGER XX1,XX2,YY1,YY2

```

```

REAL    XMN, XMX, YMN, YMX
        INSIDE, RES, FOUND, FIRST
REAL    CURX(10000), CURY(10000)

OPEN(UNIT=7, FILE=DIGTBL, IOSTAT=ERROR, STATUS='OLD',
+     FORM='FORMATTED', READONLY)

DIGTBL = '[-MAPS]PHYS.TBL'
FOUND = .FALSE.
FIRST = .TRUE.

30  READ(7, 75, END=60) FILNAM, XX1, XX2, YY1, YY2, LNO, PTFIL
    INSIDE = .FALSE.

```

```

        UPPER = LEVELNO + 1
        CALL CHECKRANGES(XX1, XX2, YY1, YY2, XMN, XMX, YMN, YMX, INSIDE)

```

```

C ** IF FILE IS AT CURRENT LEVELNO OR ONE LEVEL HIGHER THEN RES IS GOOD **
IF (LNO.EQ. LEVELNO) THEN
    RES = .TRUE.
ELSE
    RES = .FALSE.
ENDIF

```

```

IF (INSIDE.EQ. .TRUE. .AND. RES.EQ. .TRUE.) THEN
    CALL READCURRENT(CURX, CURY, FILNAM, CURTOT)
    CALL PLOTPTS(CURX, CURY, CURTOT)
    FOUND = .TRUE.
    FIRST = .FALSE.
    GOTO 30
ELSE
    GOTO 30
ENDIF

```

```

C IF NO DIG FILES FOUND AND THIS IS NOT THE FIRST ITERATION AND
C WE ARE NOT PLOTTING THE PHYSICAL FEATURES LAYER, THEN PLOT
C THE WORLD OUTLINE

```

```

60  IF (FOUND.EQ. .FALSE. .AND. FIRST.EQ. .TRUE. .AND. DIGTBL.NE.
+
        FILNAM = '[-MAPS]WORLD3.OUT'
        CALL READCURRENT(CURX, CURY, FILNAM, CURTOT)
        CALL PLOTPTS(CURX, CURY, CURTOT)
    ENDIF

```

```

        CLOSE(UNIT=7)
        CLOSE(UNIT=8)

```

```

75  FORMAT(A18, 2X, 14, 1X, 14, 1X, 13, 1X, 12, 1X, 11, 1X, A18) IBROWSE MODIFICATION
76  FORMAT(A12, 1X, 14, 1X, 14, 1X, 13, 1X, 12, 1X, 11, 1X, A12, 1X, 12)

```

```

        RETURN
        END

```

```

C **** CHECKWINDOW TO SEE IF DIG FILE IS WITHIN WINDOW *****

```

```

        SUBROUTINE CHECKRANGES(XX1, XX2, YY1, YY2, XMN, XMX, YMN, YMX, INSIDE)

```

```

        REAL X1, X2, Y1, Y2, XMN, XMX, YMN, YMX
        INTEGER XX1, XX2, YY1, YY2
        LOGICAL INSIDE

```

```

        X1 = REAL(XX1)
        X2 = REAL(XX2)

```

```

        Y1 = REAL(YY1)
        Y2 = REAL(YY2)

```

```

C ** CHECK TO SEE IF ZOOM WINDOW OVERLAPS DIG FILE **

```

```

IF (XMN.LT. X2 .AND. XMN.GT. X1 .AND. YMN.LT. Y2
+   .AND. YMX.GT. Y1) THEN
    INSIDE = .TRUE.
ELSE IF (XMX.GT. X1 .AND. XMX.LT. X2 .AND.
+   YMX.GT. Y1 .AND. YMX.LT. Y2) THEN
    INSIDE = .TRUE.
ELSE IF (XMX.GT. X1 .AND. XMX.LT. X2 .AND.
+   YMN.LT. Y2 .AND. YMN.GT. Y1) THEN
    INSIDE = .TRUE.
ELSE IF (XMN.GT. X1 .AND. XMN.LT. X2 .AND.
+   YMN.GT. Y1 .AND. YMN.LT. Y2) THEN
    INSIDE = .TRUE.

```

```

C ** CHECK TO SEE IF DIG FILE COMPLETELY CONTAINED IN ZOOM WINDOW **

```

```

ELSE IF (XMN.LT. X1 .AND. XMX.GT. X2 .AND.
+   YMN.LT. Y1 .AND. YMX.GT. Y2) THEN
    INSIDE = .TRUE.

```

```

C ** CHECK T SEE IF DIG FILE INTERSECTS ZOOM WINDOW **

```

```

ELSE IF (X1.GT. XMN .AND. X1.LT. XMX .AND.
+   Y2.GT. YMN .AND. Y2.LT. YMX) THEN
    INSIDE = .TRUE.
ELSE IF (X2.GT. XMN .AND. X2.LT. XMX .AND.
+   Y2.GT. YMN .AND. Y2.LT. YMX) THEN
    INSIDE = .TRUE.
ELSE IF (X2.GT. XMN .AND. X2.LT. XMX .AND.
+   Y1.GT. YMN .AND. Y1.LT. YMX) THEN
    INSIDE = .TRUE.
ELSE IF (X1.GT. XMN .AND. X1.LT. XMX .AND.
+   Y1.GT. YMN .AND. Y1.LT. YMX) THEN
    INSIDE = .TRUE.

```

```

ELSE INSIDE = .FALSE.
ENDIF

```

```

        RETURN
        END

```

```

C ***** SET BOUNDS FOR REFERENCE BOX *****

```

```

        SUBROUTINE SETWIN(X1, X2, Y1, Y2, OX1, OX2, OY1, OY2)

```

```

        REAL X1, X2, Y1, Y2, OX1, OX2, OY1, OY2

```

```

        OX1 = X1
        OX2 = X2
        OY1 = Y1
        OY2 = Y2

```

```

        RETURN
        END

```

```

C ***** RECORD WINDOW COORDINATES *****

```

```

        SUBROUTINE RECORD_WINDOW(XMN, XMX, YMN, YMX, XREC, YREC, CURLEVEL)

```



```
REAL XMN, XMX, YMN, YMX, XREC(10), YREC(10)
INTEGER CURLEVEL, N
```

```
CURLEVEL = CURLEVEL + 1
```

```
N = 2*CURLEVEL - 1
```

```
XREC(N) = XMN
```

```
XREC(N+1) = XMX
```

```
YREC(N) = YMN
```

```
YREC(N+1) = YMX
```

```
RETURN
END
```

```
C ***** ZOOM BACK ONE LEVEL *****
```

```
SUBROUTINE GOBACK(XMN, XMX, YMN, YMX, XREC, YREC, CURLEVEL,
+ CXMN, CXMX, CYMN, CYMX)
```

```
REAL XMN, XMX, YMN, YMX, XREC(10), YREC(10)
```

```
REAL CXMN, CXMX, CYMN, CYMX
```

```
INTEGER N, CURLEVEL
```

```
CURLEVEL = CURLEVEL - 1
```

```
N = 2*CURLEVEL-1
```

```
IF (CURLEVEL .LE. 0) THEN
```

```
  XMN = CXMN
```

```
  XMX = CXMX
```

```
  YMN = CYMN
```

```
  YMX = CYMX
```

```
ELSE
```

```
  XMN = XREC(N)
```

```
  XMX = XREC(N+1)
```

```
  YMN = YREC(N)
```

```
  YMX = YREC(N+1)
```

```
ENDIF
```

```
RETURN
END
```

```
C ***** CHOOSE CONTINENT *****
C BROWSE MODIFICATION IN SUBROUTINE NAME
```

```
SUBROUTINE MAPMENU(CURRENT, CXMN, CXMX, CYMN, CYMX, CONT, PTFIL, DIGTBL)
```

```
CHARACTER*18 CURRENT, CONT, PTFIL, DIGTBL
```

```
CHARACTER*80 PROMPT, CLEAR
```

```
REAL X(5), Y(5), CXMN, CXMX, CYMN, CYMX
```

```
INTEGER CNT
```

```
CONT = 'GLOBAL VIEW'
```

```
CURRENT = 'L.MAPS\WORLD3.OUT'
```

```
DIGTBL = 'L.MAPS\TITLE.TBL'
```

```
PTFIL = 'L.MAPS\WORLD.PTS'
```

```
CXMX = 180
```

```
CXMN = -180
```

```
CYMX = 90
```

```
CYMN = -85
```

```
15 FORMAT(11)
```

```
RETURN
END
```

```
!BROWSE MODIFICATIONS
```

```
C ***** READ VECTOR DATA FILES *****
```

```
SUBROUTINE READCURRENT(CURX, CURY, CURRENT1, CURTOT)
```

```
REAL CURX(10000), CURY(10000)
```

```
CHARACTER*18 CURRENT1
```

```
INTEGER CURTOT(2000)
```

```
INTEGER ERROR, TMP, J, N, COUNT, CNT
```

```
OPEN(UNIT=6, FILE=CURRENT1, IOSTAT=ERROR, STATUS='OLD',
```

```
+ FORM='UNFORMATTED', READONLY)
```

```
IF(ERROR.GT.0) THEN
```

```
  WRITE(*,*) IOSTAT= 'ERROR', FILE= 'CURRENT1
```

```
  STOP
```

```
ENDIF
```

```
COUNT = 1
```

```
CNT = 1
```

```
C * READ DIG FILE INTO ARRAYS FOR QUICK DISPLAY *
```

```
50 READ(6, END=500) CURX(COUNT), CURY(COUNT), CURTOT(CNT)
```

```
  TMP = COUNT+CURTOT(CNT)-1
```

```
  COUNT=COUNT+1
```

```
  DO 200 J=COUNT, TMP
```

```
    READ(6) CURX(J), CURY(J)
```

```
  CNT = CNT+1
```

```
  COUNT = J
```

```
  GOTO 50
```

```
C ** END OF FILE **
```

```
500 CURTOT(CNT) = -1
```

```
  CLOSE(UNIT=6)
```

```
  RETURN
```

```
  END
```

```
C ***** READ POINT LOCATION FILES *****
```

```
SUBROUTINE READPTS(PTFIL, XMN, XMX, YMN, YMX, LEVELNO)
```

```
CHARACTER*18 PTFIL
```

```
REAL Y,X,X2,Y2,XMN,XMX,YMN,YMX
```

```
INTEGER ERROR, LEVELNO
```

```
CHARACTER*30 PLACE
```

```
CHARACTER CH
```

```
OPEN (7, FILE=PTFIL, IOSTAT=ERROR, STATUS='OLD',
```

```
+ FORM='FORMATTED', READONLY)
```

```
CALL GKSSSET_TEXT_HEIGHT(0.08)
```

```
!BROWSE MODIFICATION
```

```
499 READ(7, FMT=501, END=502) PLACE, Y, X
```

```
C ** CONVERT FROM WORLD TO VIEW PORT COORDINATES **
```

```
  X2 = X/(XMN-XMX)
```

```

Y2 = Y/(YMX-YMN)
CALL GTX(X,Y,I+1)
IF (LEVELNO - EQ. 3) THEN
  X = X + 0.05
  Y = Y + 0.05
  CALL GTX(X,Y,PLACE)
ENDIF

```

```

GOTO 499

```

```

502 CLOSE(UNIT=7)

```

```

501 FORMAT(A27,F6.2,F7.2)

```

```

RETURN
END

```

```

C ***** SET OR RESET WINDOWS *****

```

```

SUBROUTINE SETUP(CXMN,CYMX,CYMN,CYMX,XMN,YMN,YMX,FLAG)

```

```

REAL XMN,XMX,YMN,YMX,CXMN,CYMX,CYMN,CYMX
INTEGER TNR1,TNR2,TNR3,TNR4
LOGICAL FLAG

```

```

REAL BX1(5),BX2(5),BX3(5),BX4(5)
REAL BY1(5),BY2(5),BY3(5),BY4(5)

```

```

IF(FLAG.EQ. .TRUE.) GO TO 37

```

```

C ** SET UP TRANSFORMATIONS **

```

```

CALL GSWN(1,CXMN,CYMX,CYMN,CYMX)
CALL GSPV(1,0.0,0.7,0.0,0.7)

```

```

CALL GSWN(2,CXMN,CYMX,CYMN,CYMX)
CALL GSPV(2,0.7,1.0,0.7,1.0)

```

```

CALL GSWN(3,0.0,1.0,0.0,1.0)
CALL GSPV(3,0.7,1.0,0.0,0.7)

```

```

CALL GSWN(4,0.0,1.0,0.0,1.0)
CALL GSPV(4,0.0,0.7,0.7,1.0)

```

```

CALL GSWN(5,XMN,XMX,YMN,YMX)
CALL GSPV(5,0.0,0.7,0.0,0.7)

```

```

C ** DEFINE WINDOWS FOR EACH TRANSFORMATION **

```

```

37 BX1(1)=CXMN
   BY1(1)=CYMN
   BX1(2)=CXMN
   BY1(2)=CYMX
   BX1(3)=CXMN
   BY1(3)=CYMX
   BX1(4)=CXMN
   BY1(4)=CYMN
   BX1(5)=CXMN
   BY1(5)=CYMN

```

```

BX2(1)=0.0
BX2(2)=0.0
BY2(2)=1.0

```

```

BX2(3)=1.0
BY2(3)=1.0
BX2(4)=1.0
BY2(4)=0.0
BX2(5)=0.0
BY2(5)=0.0

```

```

BX3(1)=XMN
BY3(1)=YMN
BX3(2)=XMN
BY3(2)=YMX
BX3(3)=XMX
BY3(3)=YMX
BX3(4)=XMX
BY3(4)=YMN
BX3(5)=XMN
BY3(5)=YMN

```

```

CALL GKSSET_PLINE_LINEWIDTH(3.0)

```

```

C ** DRAW BOXES ON DISPLAY **

```

```

CALL GSELNT(5)
CALL GPL(5,BX3,BY3)

```

```

CALL GSELNT(2)
CALL GPL(5,BX1,BY1)

```

```

CALL GSELNT(3)
CALL GPL(5,BX2,BY2)

```

```

CALL GSELNT(4)
CALL GPL(5,BX2,BY2)

```

```

CALL GKSSET_PLINE_LINEWIDTH(1.0)

```

```

RETURN
END

```

```

C ***** DRAW VECTORS *****

```

```

SUBROUTINE PLOTPTS(CURX,CURY,CURTOT)
REAL CURX(10000),CURY(10000),X(10000),Y(10000)
INTEGER CURTOT(2000),I,COUNT,CNTR2

```

```

I = 1
COUNT = 1

```

```

DO WHILE(CURTOT(1).GT.0)

```

```

  DO 304 CNTR2 = 1,CURTOT(1)
    X(CNTR2) = CURX(COUNT)
    Y(CNTR2) = CURY(COUNT)
    COUNT = COUNT+1
    CALL GPL(CURTOT(1),Y,X)
    I = I+1
  END DO

```

```

END DO

```

```

RETURN
END

```

```

C ***** DRAW BOX ON REFERENCE MAP *****

```

```

SUBROUTINE DRAWBOX(XMN,XMX,YMN,YMX)

```

```

REAL      XMN, XMX, YMN, YMX
REAL X(5), Y(5)

X(1) = XMN
X(2) = XMX
X(3) = XMX
X(4) = XMN
X(5) = XMN

Y(1) = YMN
Y(2) = YMN
Y(3) = YMX
Y(4) = YMX
Y(5) = YMN

CALL GPL(5,X,Y)

RETURN
END

C ***** ANNOTATE SCREEN *****
SUBROUTINE TEXT(XMN,XMX,YMN,YMX,CXMN,CXMX,CYMN,CYMX,CONT,LNO)
REAL      CXMN,CXMX,CYMN,CYMX
REAL      XMN,XMX,YMN,YMX,XB1,XB2,YB1,YB2
CHARACTER*7 ONE,TWO,THR,FOUR
CHARACTER*18 CONT_LEVEL
INTEGER   SPACE,LNO

XB1 = CXMN
XB2 = CXMX
YB1 = CYMN
YB2 = CYMX

SPACE = 32

OPEN(UNIT=6, FILE='TMP', ACCESS='DIRECT', FORM='FORMATTED',
+ RECL=SPACE, STATUS='SCRATCH')

C ** CONVERT REALS INTO CHARACTERS **

WRITE(6,REC=1,FMT=78)XB1,XB2,YB1,YB2
WRITE(6,REC=2,FMT=78)XMN,XMX,YMN,YMX
WRITE(6,REC=3,FMT=80)LNO

READ(6,REC=1,FMT=79)ONE,TWO,THR,FOUR
READ(6,REC=3,FMT=81)LEVEL

CALL GSELNT(3)

CALL GSTXFP(7,2)
CALL GSCHXP(2,6)
CALL GSCHH(0.025)

C ** ANNOTATE DISPLAY **

CALL GTX(0.0,0.95,' DISPLAY INFORMATION')
CALL GTX(0.0,0.80,' REFERENCE MAP: ')
CALL GTX(0.5,0.75,CONT)
CALL GTX(0.0,0.65,' MIN MAX ')
CALL GTX(0.0,0.6,' LAT: ')
CALL GTX(0.25,0.6,THR)
CALL GTX(0.62,0.6,FOUR)
CALL GTX(0.0,0.55,' LOW: ')
CALL GTX(0.25,0.55,ONE)

```

```

CALL GTX(0.62,0.55,TWO)

READ(6,REC=2,FMT=79)ONE,TWO,THR,FOUR

CALL GTX(0.0,0.32,' ZOOM WINDOW')
CALL GTX(0.0,0.25,' MIN MAX ')
CALL GTX(0.0,0.20,' LAT: ')
CALL GTX(0.0,0.15,' LON: ')
CALL GTX(0.30,0.20,THR)
CALL GTX(0.62,0.20,FOUR)
CALL GTX(0.30,0.15,ONE)
CALL GTX(0.62,0.15,TWO)
CALL GTX(0.0,0.05,' ZOOM LEVEL: ')
CALL GTX(0.6,0.05,LEVEL)

CLOSE(UNIT=6)

CALL GSELNT(4)
CALL GSCHH(0.048)
CALL GSCHXP(1.0)

CALL GTX(0.1,0.9,' UCSB BROWSE - WORLD PLOT')
CALL GTX(0.10,0.75,' USER MENU')

C BROWSE MODIFICATIONS--THESE PROMPTS ARE NOW IN MAPCHOICE SUBROUTINE
C CALL GTX(0.10,0.6,' 1: ZOOM')
C CALL GTX(0.10,0.5,' 2: UNZOOM')
C CALL GTX(0.10,0.4,' 3: GO BACK TO LAST ZOOM')
C CALL GTX(0.10,0.3,' 4: RETURN WINDOW TO BROWSE')
C CALL GTX(0.10,0.12,' PLEASE POINT TO SELECTION:')

78 FORMAT(4F7.2)
79 FORMAT(4A7)
80 FORMAT(11)
81 FORMAT(A2)

RETURN
END

C ***** CONVERT MOUSE INPUT TO WORLD COORDINATES *****
SUBROUTINE CHOOSEWIN(X1,X2,Y1,Y2,STAT,TNR,DISP,XMN,XMX,YMN,YMX)

CHARACTER CH
DATA XN/0.0/,XX/0.7/,YN/0.0/,YX/0.7/

CHARACTER*80 CHR
CHARACTER*6 ONE,TWO,THREE,FOUR
REAL X1,X2,Y1,Y2,XMN,XMX,YMN,YMX
REAL XH(2),YH(2),XV(2),YV(2)
INTEGER STAT,TNR2,DISP

XH(1) = XMN
XH(2) = XMX
YV(1) = YMN
YV(2) = YMX

TNR2=1

C * GET FIRST MOUSE INPUT *
C CALL GINLC(DISP,1,0.0,50.0,TNR,1,XN,XX,YN,YX,CHR,1)
CALL GSLCM(DISP,1,0,1)
CALL GRQLC(DISP,1,STAT,TNR2,X1,Y1)
X1=(X1*(XMX-XMN)+1.4)*XMN
Y1=(Y1*(YMX-YMN)+1.4)*YMN

XV(1) = X1

```

```
XV(2) = X1
YH(1) = Y1
YH(2) = Y1
```

```
C *      DRAW CROSSHAIR
CALL GKSSSET_PLINE_LINETYPE(4)
CALL GPL(2,XV,YV)
CALL GPL(2,XH,YH)
```

```
C *      GET SECOND MOUSE INPUT
CALL GRQLC(DISPLAY,1,STAT,TNR2,X2,Y2)
X2=(X2*(XHX-XHN)+1.4)+XHN
Y2=(Y2*(YHX-YHN)+1.4)+YHN
```

```
XV(1) = X2
XV(2) = X2
YH(1) = Y2
YH(2) = Y2
```

```
C *      DRAW CROSSHAIR
CALL GPL(2,XV,YV)
CALL GPL(2,XH,YH)
CALL GKSSSET_PLINE_LINETYPE(1)
```

```
RETURN
END
```

```
C ***** UPDATE ZOOMED COORDINATES *****
```

```
SUBROUTINE SETBOUNDS(X1,X2,Y1,Y2,XHN,XHX,YHN,YHX)
```

```
REAL X1,X2,Y1,Y2
```

```
XHN = X1
XHX = X2
YHN = Y1
YHX = Y2
```

```
RETURN
END
```

```
C ***** PROCESS ZOOM OUTLINE *****
```

```
SUBROUTINE ZOOM(X1,X2,Y1,Y2,XHN,XHX,YHN,YHX)
```

```
REAL X1,Y1,X2,Y2,XHN,XHX,YHN,YHX
REAL TMPX,TMPY,AVE,RANGE,DELTX,DELTY
INTEGER PORT
```

```
PORT = 5
```

```
DELTX = XHX-XHN
DELTY = YHX-YHN
```

```
C *      ORDER X,Y PAIRS TO MIN,MAX *
```

```
IF(X1.GT.X2) THEN
  TMPX = X1
  X1 = X2
  X2 = TMPX
ENDIF
```

```
IF(Y1.GT.Y2) THEN
  TMPY = Y1
  Y1 = Y2
  Y2 = TMPY
ENDIF
```

```
C *      LENGTHEN SHORTER END OF ZOOMED BOX *
C *      TO RETAIN PROPER ASPECT RATIO *
```

```
  TMPX = (X2-X1)/DELTX
  TMPY = (Y2-Y1)/DELTY
```

```
IF(TMPX.GT.TMPY) THEN
  AVE = (Y1+Y2)/2
  RANGE = DELTY/2 * TMPX
  Y1 = AVE - RANGE
  Y2 = AVE + RANGE
ELSE
  AVE = (X1+X2)/2
  RANGE = DELTX/2 * TMPY
  X1 = AVE - RANGE
  X2 = AVE + RANGE
ENDIF
```

```
C *      SET NEW TRANSFORMATION *
CALL GSWN(5,X1,X2,Y1,Y2)
CALL GSWP(5,0.0,0.7,0.0,0.7)
CALL GSELNT(5)
```

```
RETURN
```

```
END
```

```
C ***** RESET TRANSFORMATION FOR ENTIRE CONTINENT *****
```

```
SUBROUTINE UNZOOM(XHN,XHX,YHN,YHX,CXHN,CXHX,CYHN,CYHX)
```

```
REAL XHN,XHX,YHN,YHX,CXHN,CXHX,CYHN,CYHX
```

```
XHN = CXHN
XHX = CXHX
YHN = CYHN
YHX = CYHX
```

```
RETURN
END
```

```
subroutine menu(menuid)
```

```
C*****
C Menu Subroutine for Browse
C selects menu display option based on termtype
C and calls appropriate subroutine
C
C author: david stoms
C date : 01/22/88
C*****
```

```
C declarations
```

```
common /term/ termtype
integer termtype
character menuid*10
```

```
C if terminal is vt100 type, use alphanumeric mode
if (termtype.eq.1) then
  call menuvt(menuid)
  goto 9999
C else use tektroniks mode
else if (termtype.eq.2) then
```

```

      call menutek(menuid)
      goto 9999
    else
      call menuascii(menuid)
    end if
  9999 return
end

  subroutine menuascii(menuid)
*****
c Menu Display Subroutine
c reads menuXX.txt file for menuid and prints to any
c ASCII terminal
c
c author: david stoms
c date : 01/22/88
c revised: 06/30/88
c
c
c declarations
common /vers/version,revdate
character revdate*8,version*3
integer I,nolines,ios,length
character menuid*10,string*40,filename*22,title*72
character esc*3
call clearscreen
c open menu text file and read text
filename(1:7) = '[.text]'
filename(8:17) = menuid
open(unit = 7,file = filename,status = 'old',err = 9005)
c read header and continue
read (7,9011) string
read (7,9012) nolines
read (7,9011) title
do 10 J= 72,1,-1
  if (title(J:J).ne. ' ') then
    length = (80 - J)/2
    goto 20
  end if
10 continue
20 print 9014,version,revdate
print 9015,title
c print title to audit.log file
write(20,9020)title
c print menu to terminal
5 do 50 I = 1, nolines
  read (7,9011) string
  if (string(1:1) .eq. '-') then
    print 9003
  else
    print 9001,string
  end if
50 continue
close(unit=7)
print 9008

```

```

100 return
9001 format(' ',10X,A)
c print blank line when ~ read in text file
9003 format(' ')
9008 format(' ',/, ' Enter choice number: ', $)
9011 format(A)
9012 format(12)
9014 format(' ',/, 5X,'UCSB-BROWSE',13X,'Build ',
2 A3,13X,'Revised ',A8)
9015 format(' ',/, 5X,A)
9020 format(' ',4X,A)
9005 print 9006,ios,menuid
9006 format(' ',/, -ERROR- ',12, ',A10,' file. ')
2 goto 100
9999 end

  subroutine menutek(menuid)
*****
c Menu Display Subroutine for Tek 4010 Screen
c reads menuXX.txt file for menuid and prints to terminal
c with direct cursor positioning
c
c author: david stoms
c date : 12/01/87
c revised: 06/30/88
c
c
c declarations
*****
INCLUDE 'SYSSLIBRARY:GKSDDFS.BND'
common /vers/version,revdate
character revdate*8,version*3
integer I,nolines,ios,length
character menuid*10,string*40,filename*22,title*72
REAL X(5),Y(5),line
INTEGER CNT
call clearscreen
c open menu text file and read text
filename(1:7) = '[.text]'
filename(8:17) = menuid
open(unit = 7,file = filename,status = 'old',err = 9005)
c draw box around menu
CALL GSWN(7,0.0,1.0,0.0,1.0)
CALL GSVP(7,0.2,0.9,0.1,0.8)
CALL GSELNT(7)
X(1)=0.0
X(2)=0.0
X(3)=1.0
X(4)=1.0
X(5)=0.0
Y(1)=0.0

```

```

Y(2)=1.0
Y(3)=1.0
Y(4)=0.0
Y(5)=0.0

```

```

c goto heavy line width
CALL GSLWSC(4.0)
CALL GPL(5,Y,X)

```

```

c read header and continue
read (7,9011) string
read (7,9012) nlines
read (7,9011) title

```

```

CALL GSCHXP(1.0)
CALL GSTXFP(7.0)
CALL GSCHH(0.025)
CALL GTX(0.10,0.90,'UCSB-BROWSE')
CALL GTX(0.10,0.84,'Build')
CALL GTX(0.32,0.84,'version')
CALL GTX(0.50,0.84,'Revised')
CALL GTX(0.75,0.84,'revdate')
CALL GTX(0.30,0.75,'title')

```

```

c print title to audit.log file
write(20,9020)title

```

```

c print menu to terminal

```

```

line = 0.8
5 do 50 I = 6, nlines + 5
  read (7,9011) string
  if (string(1:1) .eq. '-') then
    line = line - .1
  else
    line = 1.0 - (I * 0.05)
  CALL GTX(0.2,line,string)
end if

```

```

50 continue
CALL GTX(0.40,0.10,'ENTER CHOICE NUMBER: ')
close(unit=7)
goto 9999

```

```

9011 format(A)
9012 format(I2)
9020 format(' ',4X,A)

```

```

c error procedure
9005 print 9006,ios,filename
9006 format(' ',12,'-ERROR- ',12,' file.')
2

```

```

9999 continue
return
end

```

```

subroutine menuvt(menuid)

```

```

*****
c Menu Display Subroutine
c reads menuX.txt file for menuid and prints to terminal
c with direct cursor positioning in vt100 mode
c
c
c author: david stoms
c date : 11/19/87
c revised: 06/30/88

```

```

c
c*****

```

```

c declarations

```

```

common /vers/version,revdate
character revdate*8,version*3
integer I,nlines,ios,length
character menuid*10,string*40,filename*22,title*72
character esc*3

```

```

call clearscreen

```

```

esc(1:1) = ' '
esc(2:2) = char(27)
esc(3:3) = '['

```

```

c open menu text file and read text

```

```

filename(1:7) = '[.text]'
filename(8:17) = menuid
open(unit = 7,file = filename,status = 'old',err = 9005)

```

```

c read header and continue
read (7,9011) string
read (7,9012) nlines
read (7,9011) title
do 10 J= 72,1,-1
  if (title(J:J) .ne. ' ') then
    length = (80 - J)/2
    goto 20
  end if

```

```

10 continue
20 print 9002
print 9014,esc,version,revdate,esc
print 9015,esc,esc,length,title,esc
print 9002
c print title to audit.log file
write(20,9020)title

```

```

c print menu to terminal
5 do 50 I = 6, nlines + 5
  read (7,9011) string
  if (string(1:1) .eq. '-') then
    print 9003,esc,I,esc,I
  else
    print 9001,esc,I,esc,I,string,esc,I
  end if

```

```

50 continue
close(unit=7)
print 9002
print 9008

```

```

100 return

```

```

9001 format(' ',A3,I2.2,';2H|',A3,I2.2,';25H|',A3,I2.2,';79H|')
9002 format(' ',1X,|-----|)
2

```

```

9003 format(' ',A3,I2.2,';2H|',A3,I2.2,';79H|')
9005 print 9006,ios,menuid
9006 format(' ',12,'-ERROR- ',12,
2
' can not open ',A10,' file.')
goto 100
9008 format(' ',/,',', Enter choice number: ',)
9011 format(A)
9012 format(I2)

```

```

9014 format('A3,13;2H',5X,'UCSB-BROWSE',15X,'Build ',
          A3,13X,'revised ',A8,A3,'5;79H',)
9015 format('A3,14;2H',A3,'4;12.2',H,A30,A3,'4;79H',)
9020 format(' ',4X,A)

9999 end

subroutine messages
*****
c Browse Messages Subroutine
c called by mainmenu subroutine
c allows user to leave comments on BROWSE testbed
c and to read messages from system
c
c author: david stoms
c date : 12/07/87
c revised: 07/05/88
*****
c declarations
common /userdata/username,institution,disipline,hardware,
2 access
integer choice
character menuid*10,string*75,username*40,institution*60,
2 today*9,nov*8,disipline*40,hardware*30,access*20,
3 sysmsg*75
c ask user if they want to read or write messages *****
1 menuid = 'menu14.txt'
call menuc(menuid)
call getchoice(choice)

c check for legal values
if ( choice.lt. 1 .or. choice .gt. 3) then
print 9200
call pause
goto 1
end if
call clearscreen
goto (100,200,9999),choice

c writing messages *****
100 continue
write(20,*) ' User leaving message'
open(unit=7, file = 'usermsg.txt', access = 'append',
2 status = 'unknown', err = 9910)
c 2 recl=512,status = 'unknown',err = 9910)

print 9001
call date(today)
call time(now)
write(7,9004)username,institution,disipline,today,now
print 9005

do 50 I=1,500
print 9002
read(*,9006) string
if (string.eq.' ') goto 150
write(7,9007) string
50 continue

150 write(7,9022)

```

```

close(unit=7)
print 9008
call pause
goto 1

c read messages from system *****
200 continue
write(20,*) ' User reading messages'
open(unit=7, file = 'sysmsg.txt', status = 'old', err = 9920)
do 250 j = 1,500
call clearscreen
do 275 k = 1,16
read (7,9010)sysmsg
if(sysmsg(1:1).eq.' ') then
call pause
goto 1
end if
print 9011,sysmsg
275 continue
250 call pause
250 continue

c format statements *****
9001 format(' ',//,30X,'UCSB-BROWSE',//
2 5X,'To send comments to the BROWSE Testbed Facility, ',//,
3 5X,'simply type in your remarks.',//,
4 5X,'When finished, type a period (".") on a line by',//,
5 5X,'itself.',)
9002 format(' ',3X,' :$)
9004 format(' ',//,5X,'Comment on UCSB-BROWSE Testbed',//,
2 5X,'User: ',A40,/,
3 5X,'Institution: ',A60,/,
4 5X,'Disipline: ',A40,/,
5 5X,'Date: ',A9,10X,'Time: ',A8,/)
9005 format(' ',//,5X,'Enter Message: ')
9006 format(A75)
9007 format(' ',4X,A75)
9008 format(' ',//,5X,'Thanks for your feedback.')
c9009 format(A512)
9010 format(A75)
9011 format(' ',4X,A75)
9022 format(' ',//,5X,'*****',//)
9200 format(' ',//,' Choice must be between 1 and 3.',/)

c can't open usermsg file for writing comments
9910 print 9915,ios
call pause
goto 1
9915 2 format(' ', '-ERROR-',12/, 'Can not open usermsg.txt',
2 ' ', file requested in messages subroutine')

c can't open sysmsg file...no messages today
9920 print 9925
call pause
goto 1
9925 format(' ', ' ...no messages for you today...',/)

9999 return
end
subroutine messages
*****
c

```

```

c Browse Messages Subroutine
c called by mainmenu subroutine
c allows user to leave comments on BROWSE testbed
c and to read messages from system
c
c author: david stoms
c date : 12/07/87
c revised: 07/05/88
c *****
c declarations
common /userdata/username,institution,discipline,hardware,
2 access
integer choice
character menuid*10,string*75,username*40,institution*60,
2 today*9,now*8,discipline*40,hardware*30,access*20,
3 sysmsg*75
c ask user if they want to read or write messages *****
1 menuid = 'menuid.txt'
call menu(menuid)
call getchoice(choice)
c check for legal values
if ( choice.lt. 1 .or. choice .gt. 3) then
print 9000
call pause
goto 1
end if
call clearscreen
goto (100,200,9999),choice
c writing messages *****
100 continue
write(20,*) ' User leaving message'
open(unit = 7, file = 'usermsg.txt', access = 'append',
2 status = 'unknown', err = 9910)
c 2 rec1=512,status = 'unknown',err = 9910)
print 9001
call date(today)
call time(now)
write(7,9004)username,institution,discipline,today,now
print 9005
do 50 i=1,500
print 9002
read (*,9006) string
if (string .eq. '.') goto 150
write(7,9007) string
50 continue
150 write(7,9022)
close(unit=7)
print 9008
call pause
goto 1
c read messages from system *****
200 continue
write(20,*) ' User reading messages'

```

```

open(unit = 7, file = 'sysmsg.txt', status = 'old', err = 9920)
do 250 j = 1,500
call clearscreen
do 275 k = 1,16
read (7,9010)sysmsg
if(sysmsg(1:1) .eq. '.') then
call pause
goto 1
end if
print 9011,sysmsg
275 continue
250 continue
c format statements *****
9001 format(' ',//,30X,'UCSB-BROWSE',//
2 5X,'To send comments to the BROWSE Testbed Facility',//,
3 5X,'simply type in your remarks.',//,
4 5X,'when finished, type a period (".") on a line by',//,
5 5X,'itself.')
```

```

9002 format(' ',//,3X,':')
9004 format(' ',//,5X,'Comment on UCSB-BROWSE Testbed',//,
2 5X,'User: ',A40,/,
3 5X,'Institution: ',A60,/,
4 5X,'Discipline: ',A40,/,
5 5X,'Date: ',A9,10X,'Time: ',A8,/)
9005 format(' ',//,5X,'Enter Message: ')
9006 format(A75)
9007 format(' ',//,4X,A75)
9008 format(' ',//,5X,'Thanks for your feedback.')
```

```

c9009 format(A512)
9010 format(A75)
9011 format(' ',//,4X,A75)
9022 format(' ',//,5X,'*****',//)
2 *****
9200 format(' ',//,' Choice must be between 1 and 3.',//)
c can't open usermsg file for writing comments
9910 print 9915,ios
call pause
goto 1
9915 2 format(' ',//,'-ERROR-',12,'Can not open usermsg.txt',
c ' file requested in messages subroutine')
```

```

c can't open sysmsg file....no messages today
9920 print 9925
call pause
goto 1
9925 format(' ',//, ...no messages for you today....',//)
9999 return
end
subroutine pause
c *****
c Browse Pause Subroutine
c called by many browse subroutines
c pauses after displaying a message or output
c and waits for carriage return before continuing
c
c author: david stoms
c date : 08/13/87
c revised: 02/10/88

```



```

end if
9999 return

c format statements *****
9002 format(' ',2X,A8,' relation not found. '//)
9003 format(' ',4X,'No records were retrieved in the last query.',
          2 //4X,'The results of the previous query are',
          3 ' restored.')
end

subroutine printrow(relname)
c*****
c Browse Printrow Subroutine
c called by many browse subroutines
c uses rimrel and ringrel to find and print
c the number of tuples in the projected relation
c
c author: david stoms
c date : 08/13/87
c revised: 07/28/88
c*****
c declarations
include '[stoms.src]params.blk'

common /rimcom/ rimstat,unit,status
integer numrows,numatts,rimstat,unit,status
real*8 relname,lastmod,rname
character*74 errmsg
logical rpw,mpw

c if no query made yet, don't print whole database
if (ptr.eq. 0) then
  print *, '...no records have been retrieved yet...'
  call pause
  return
end if

c look thru relations until name matches *****
10 call rimrel
call ringrel(rname,rpw,mpw,lastmod,numatt,numrows)
c while more relations remain, look at next relation
if(rimstat.ne. -1) then
  c if its the desired relation, print numrows
  if (rname.eq. relname) then
    if(numrows.eq. 0) then
      call zerohits(relname,numrows)
      goto 9999
    end if
    print 9001,relname,numtuples = numrows
    call pause
    goto 9999
  end if
  print 9002,relname
  call ringmsg(errmsg)
  print *,errmsg
else
  print 9003,relname
  call ringmsg(errmsg)
  print *,errmsg

```

```

end if
9999 continue
return

c format statements *****
9001 format(' ',16X,'Current number of records in ',A8,' = ',
          2 I3,/)
9002 format(' ',2X,A8,' relation not found. '//)
end

c
c
c subroutine zerohits(relname,numrows)
c*****
c Browse Zerohits Subroutine
c called by locproj and printrow subroutines
c checks number of records in current temporary relation
c if number is zero, pointer is decremented by 1 and
c temp relation is removed
c
c author: david stoms
c date : 07/21/88
c revised: 07/28/88
c*****
c declarations
include '[stoms.src]params.blk'

common /rimcom/ rimstat,unit,status
integer numrows,numatts,rimstat,unit,status
real*8 relname,lastmod,rname
character*74 errmsg
logical rpw,mpw

c look thru relations until name matches *****
10 call rimrel
call ringrel(rname,rpw,mpw,lastmod,numatt,numrows)
c while more relations remain, look at next relation
if(rimstat.ne. -1) then
  c if its the desired relation, print numrows
  if (rname.eq. relname) then
    if(numrows.eq. 0) then
      print 9003
      call rimrem(relname)
      ptr = ptr - 1
      call pause
      goto 9999
    end if
    goto 10
  end if
  print 9002,relname
  call ringmsg(errmsg)
  print *,errmsg
end if
9999 return

```

```

c format statements *****
9002 format(' ',/2X,A8,' relation not found. '///)
9003 format(' ',/4X,'No records were retrieved in the last query.',
3      /,4X,'The results of the previous query are',
      /, restored.'/)
      end
2      subroutine proj(newrel,oldrel,attrib,kalen,
      clause,kclen)
*****
c      Browse Proj Subroutine
c      called by many browse subroutines
c      projects a temporary relation in rim from an original
c      relation based on the user specified query attributes
c      and displays number of tuples selected in query
c
c      author: david stoms
c      date : 11/06/87
c*****
c declarations
      common /rimcom/ rimstat,unit,status
      integer kalen,kclen,numatt,numrows,rimstat,unit,status
      real*8 newrel,oldrel,attrib(*),klause(*)
      logical rname,lastmod
      logical rpw,mpw
      character*74 errmsg
c call to rimproj*****
      call rimproj(newrel,oldrel,attrib,kalen,klause,kclen)
c check rimstat and print error message
if (rimstat.ne.0) then
      call rimmsg(errmsg)
      print *,errmsg
      call pause
      goto 9999
end if
c find relation in data base and print number of tuples
      call printrow(newrel)
c format statements *****
9999 continue
return
end
      subroutine query(menuid)
*****
c      Browse Query Subroutine
c      called from mainmenu subroutine with menu11.txt
c      calls menu.sub for query menu, reads choice
c
c      author: david stoms
c      date : 12/01/87
c      revised: 02/10/88
c*****

```

```

c declarations
      integer choice
      character menuid*10
c print menu to terminal
1      call menu(menuid)
      call getchoice(choice)
c check for legal values
if (choice.lt.1 .or. choice.gt.6) then
      print 9000
      call pause
      goto 1
endif
      goto(100,200,300,400,500,600),choice
c query catalog option
100 menuid = 'menu21.txt'
      call catalog(menuid)
      goto 9999
c query directory option
200 menuid = 'menu22.txt'
      call directory(menuid)
      goto 9999
c query user data base option
300 continue
      menuid = 'menu23.txt'
      call user(menuid)
      call clearscreen
      print *, 'Sorry, USER data base is not currently available.'
      call pause
      menuid = 'menu11.txt'
      goto 1
c help menu
400 continue
      call help
      goto 1
c return to main menu
500 goto 9999
600 call quit
9000 format(' ', 'Choice must be between 1 and 6')
c 9070 format(' ', 'Enter choice #: ', $)
9999 return
end
      subroutine quit
*****
c      Browse Quit Subroutine
c      displays goodbye banner
c
c      author: david stoms
c      date : 11/23/87
c*****

```



```
c assign names to newrel,oldrel
      oldrel = tempname(ptr)
      newrel = tempname(ptr + 1)
```

```
c select all attributes
```

```
      attrib(1) = 1H0
      kalen = 0
```

```
c increment pointer
      ptr = ptr + 1
      return
end
```

```
subroutine setterm
```

```
*****
c Browse Setterm Subroutine
c called from main program to get the user's terminal/emulator *
c type for setting proper clearscreen escape codes *
c and opens gks if tek mode selected *
c
c author: david stoms *
c date : 01/22/88 *
c revised: 03/11/88 *
c *****
```

```
c declarations
```

```
include 'sys$library:gksdefs.bnd'
```

```
common /term/ termtype
integer termtype,disp,syserror,type
data disp/1/,syserror/0/,type/72/
```

```
c ask for users terminal/emulator type *****
```

```
10 print 9001
print 9070
call getchchoice(termtype)
c check for legal values
if (termtype.lt. 1 .or. termtype.gt. 4) then
  print 9200
  call pause
  goto 10
endif
```

```
c if vt mode, just write to audit.log
if (termtype.eq. 1) then
```

```
  write(20,*) ' VT100 terminal'
```

```
c if tek mode, open gks stuff
else if (termtype.eq. 2) then
```

```
  write(20,*) ' TEK 4010 terminal'
  call gopks(syserror)
  call gopwk(disp,gks$konid_default,type)
  call gacwk(disp)
```

```
c gds needed for gks.level 2c upgrade
call gds(disp,0,0)
```

```
c if ascii mode, tell audit.log about it
else if (termtype.eq. 3) then
```

```
  write(20,*) ' ASCII terminal'
c otherwise quit browse program
else if (termtype.eq. 4) then
```

```
      write(20,*) ' Quit BROWSE at setterm'
      stop
end if
```

```
c format statements *****
```

```
9001 format(' ',4X,'What terminal emulator type are you using?',//
```

```
2 15X,'1) VT100 series',//
```

```
3 15X,'2) TEK 4010/4014',//
```

```
4 15X,'3) ASCII terminals',//
```

```
5 15X,'4) Quit BROWSE',//
```

```
9070 format(' ',4X,'Enter choice number: ',//
```

```
9200 format(' ',4X,'Choice must be between 1 and 4')

```

```
9999 return
end
```

```
subroutine transmode(mode)

```

```
*****
c Browse Transmode Subroutine
c asks user what connection type--modem, SPAN, or ARPANET--*
c they have and thus which transfer mode to use *
c
c author: david stoms *
c date : 05/11/88 *
c revised: 05/11/88 *
c *****
```

```
c declarations
```

```
character menuid*10
integer mode
```

```
menuid = 'menu49.txt'
call menu(menuid)
```

```
call getchchoice(mode)
```

```
c check for legal values
if (mode.type.lt. 1 .or. mode.type.gt. 3) then
  print 9200
  call pause
  goto 1
end if
```

```
c format statements *****
```

```
9200 format(' ',4X,'Choice must be between 1 and 3.',//)
```

```
9999 return
end
```

```
subroutine userdata
```

```
*****
c Browse Userdata Subroutine
c called by main subroutine *
c gets username, institution, and discipline for user stats *
c and for messages subroutine *
c
c author: david stoms *
c date : 12/07/87 *
c revised: 06/30/88 *
c *****
```

```
c declarations
```

```

common /userdata/username,institution,discipline,hardware,
2      access
common modetype

character username*40,institution*60,discipline*40,
2      hardware*30,access*20
integer modetype

      call clearsreen

10 print 9001
   print 9002
   read (*,9010)username
   if (username .eq. ' ') then
      print 9200
      goto 10
   end if
   print 9003
   read (*,9011)institution
   print 9004
   read (*,9010)discipline
   print 9006
   read (*,9012)hardware
   call transmode(modetype)
   if(modetype .eq. 1) then
      access = 'DIAL-UP/MODEM'
   else if (modetype .eq. 2) then
      access = 'SPAN/DECNET'
   else
      access = 'ARPANET/NSFNET'
   end if

```

```

c write userdata to audit.log file
write(20,9005)username,institution,discipline,
2      hardware,access

```

```

c format statements *****
9001 format(' ',15X,'Please enter the following information. ')
9002 format(' ',//,5X,'Name: ', $)
9003 format(' ',4X,'Institution: ', $)
9004 format(' ',4X,'Discipline: ', $)
9005 format(' ',//,5X,'User: ',A40,/,
2      5X,'Institution: ',A60,/,
3      5X,'Discipline: ',A40,/,
4      5X,'Hardware: ',A30,/,
5      5X,'Access: ',A20,/,
9006 format(' ',4X,'Hardware (e.g., IBM-PCAT): ', $)
9010 format(A40)
9011 format(A60)
9012 format(A30)
9200 format(' ',//, ' Please enter your name.',/)

9999 return
      end

```

```

Dec Command Language: BLD_LIB.COM

```

```

$! Create Browse Library of Object Code
$! write sys$output "Compile Source Code"
$ fortran main
$ fortran htoc
$ fortran kernspwn

```

```

$ fortran listline
$ fortran mainmenu
$ fortran query
$ fortran catalog
$ fortran menu
$ fortran setterm
$ fortran dispans
$ fortran menutek
$ fortran menuvnt
$ fortran menuasci
$ fortran getchoice
$ fortran transmode
$ fortran listplat
$ fortran listsens
$ fortran listloc
$ fortran listtime
$ fortran quit
$ fortran clearscr
$ fortran listcloud
$ fortran proj
$ fortran printrow
$ fortran dispatts
$ fortran relselct
$ fortran dispsort
$ fortran cleanup
$ write sys$output "ABOUT HALFWAY DONE"
$ fortran locproj
$ fortran mapwd
$ fortran pause
$ fortran messages
$ fortran help
$ fortran banner
$ fortran userdata
$ fortran director
$ fortran listcont
$ fortran listcount
$ fortran listnode
$ fortran listform
$ fortran dispatt2
$ fortran relsct2
$ fortran printrow2
$ fortran dispans2
$ fortran listplat2
$ fortran listsens2
$ fortran bantek
$!
$ write sys$output "Create Browse Library"
$!
$! library BROWSE.OLB *.obj
$!
$ write sys$output "BROWSE LIBRARY COMPLETE"
$!
$! get rid of separate object files
$ delete *.obj.*
$ ESC[0,8]
$ write sys$output "ESC[7m"
$ write sys$output " ROCK N' ROLL"
$ write sys$output "ESC[0m"
Dec Command Language: UPDATE.COM
$!UPDATE.COM
$!
$! UPDATES BROWSE OBJECT LIBRARY
$! one file at a time
$!

```

```
$ write sys$output "Let's update the BROWSE object library!"
$ inquire FILE "Which File?"
```

```
$!
$ fortran 'FILE'
$ inquire CHOICE "Replace object in Browse library? [Y/N]?"
$ if CHOICE .eq. "n" then goto END_JOB
$ library/replace BROWSE.OLB 'FILET'
$ inquire CHOICE "Link new executable file? [Y/N]?"
$ if CHOICE .eq. "n" then goto END_JOB
$ link/executable={stoms}browse.exe -
    main.obj,browse/lib, -
    $disk2:trimjrimlib/li, -
    sys$library:gksforbnd/library
$ set file {stoms}browse.exe /pro = (w:re)
$! don't delete main.obj or else link won't work
$ if FILE .eqs. "MAIN" then goto END_JOB
$ FILE = FILE + ".OBJ.*"
$ delete 'FILE'
$END JOB:
$ ESC[0,8] = %X18
$ write sys$output ""ESC'[7m"
$ write sys$output "      ROCK N' ROLL"
$ write sys$output ""ESC'[0m"
```

```
Dec Command Language: DATA.COM
```

```
$!data.com file to compile/link dataprompt for browse
$ fortran/object=data dataprompt+clearscr+pause
$ link data
```

```

/* MAPWD SOURCE CODE LISTING FOR TRANSLATION FROM
   FORTRAN77 IN MICROVAX II ENVIRONMENT USING GKS
   TO TURBOC FOR IBM PC-AT USING TURBOC GRAPHICS
*/

```

```

#include <dos.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <stdarg.h>

#include <stdio.h>
#include <fcntl.h>
#include <sys\stat.h>
#include <string.h>
#include <io.h>
#include <alloc.h>

#include <graphics.h>

#define ESC 0x1b
#define TRUE 1
#define FALSE 0
#define PI 3.14159
#define ON 1
#define OFF 0

/* Define some handy constants */
/* Define some handy constants */
/* Define a value for PI */
/* Define some handy constants */
/* Define some handy constants */

```

```

/***** declarations for main program *****/

```

```

int tnr,choice,curtot[1000];
int levelno,curlevel,curview,dum,comp;
char *prompt,*digtbl,*cont,*ptfil,*refer;
float x1,x2,y1,y2,curx[4000],cury[4000];
float xmin,ymax,ymn,ymx,cxmn,cymn,cymx;
float ox1,ox2,oy1,oy2,xrec[10],yrec[10];
float xrange,yrange,xratio,yratio;

```

```

int zoomed,flag,done;

```

```

int tnr1,tnr2,tnr3,tnr4;
struct viewporttype view;

```

```

/***** declarations for turbo C graphics *****/

```

```

int g_driver,g_mode,g_error,nscale;

int viewnum;
int one_xmin,one_ymax,two_xmin,two_ymax,three_xmin,three_ymax;
int one_ymn,one_ymx,two_ymn,two_ymx,three_ymn,three_ymx;
void *buffer;
void *cursor;

```

```

FILE *fdopen(int handle, char *type);
/* mapwd function prototypes */

```

```

void initgrf(void);
void mainmenu(void);
void setwin(void);
void readcurrent(void);
void setup(void);

```

```

void plotpts(void);
void drawbox(void);
void setlevel(void);
void plot_digfls(void);
void readpts(void);
void text2(int x, int y);
void update(int x, int y);

void choosewin(void);
void zoom(void);
void setbounds(void);
void record_window(void);

void unzoom(void);
void goback(void);

int checkranges(float *xx1,float *xx2,float *yy1,float *yy2);

/* turboc graphics utilities routines */

void setview(int *viewnum);
void def_scr(void);
int transform(float oldx);
int transformy(float oldy);

```

```

/***** BEGIN CODE *****/

```

```

int main ()
{

```

```

/* begin code */

```

```

/* allocate space for dynamic vars */

```

```

digtbl = (char *) malloc(20);
cont = (char *) malloc(20);
ptfil = (char *) malloc(20);
refer = (char *) malloc(20);

```

```

initgrf();
getviewsettings(&view);
def_scr();

```

```

/* set up main menu and defaults */

```

```

done = FALSE;
zoomed = FALSE;

```

```

while(done != TRUE)
{

```

```

    mainmenu();
    if (zoomed == FALSE)
    {
        xmin = cxmn;
        xmax = cymx;
        ymn = cymn;
        ymx = cymx;
    }

```



```

    curlevel = 0;
    levelno = 4;

    setwin();
    printf("Reading data please wait...\n");
    readcurrent();
}

viewnum = 4;
setview(&viewnum);
clearviewport();
setup();
viewnum = 3;
setview(&viewnum);

setlevel();

if (((refer == "world3.out") && (levelno == 1)) || zoomed == FALS
----->E)
    else {
        plotpts();
        plot_digfils();
        if (levelno == 3)
        {
            strcpy(digtbl,"phys.tbl");
            viewnum = 3;
            setview(&viewnum);
            plot_digfils();
            strcpy(digtbl,"tile.tbl");
        }

        /* plot points */
        if (ptfil != "world.pts")
            readpts();

        /* annotate display */
        strcpy(cont,refer);
        text();

        /* choose zoom, unzoom, main menu */
        choice = getch();

        switch (choice)
        {
            case 49 : {
                viewnum = 3;
                setview(&viewnum);
                choosewin();
                setwin();
                zoom();
                setbounds();
                record_window();
                zoomed = TRUE;
                break;
            }
            case 50 : {
                clearviewport();
                unzoom();
                setwin();
                zoomed = FALSE;
                levelno = 1;
                curlevel = 0;
                break;
            }
        }
    }
}

```

```

    }
    case 51 : {
        goback();
        setwin();
        break;
    }
    case 52 : {
        done = TRUE;
    }
}

closegraph();

free(digtbl);
free(cont);
free(refer);
free(ptfil);

printf("see ya\n");
}

/***** mapud function definitions *****/
void initgrf()
{
    /* initialize graphics display */
    detectgraph(&g_driver,&g_mode);
    if (g_driver < 0)
    {
        printf("No graphics hardware detected - Bailing out!\n");
        exit(1);
    }

    initgraph(&g_driver,&g_mode,"");
    g_error = graphresult();
    if (g_error < 0)
    {
        printf("intigraph error: %s.\n",grapherrormsg(g_error));
        exit(1);
    }

    void readcurrent()
    {
        int tmp,j,n,count,cnt;
        int numscan,dum,handle;
        FILE *datfil;

        handle = open(refer,O_RDONLY);
        datfil = fdopen(handle,"r");

        if (datfil == NULL){
            printf("Cannot open %s\n",refer);
            getch();
            exit(1);
        }
        count = 1;
        cnt = 1;
    }
}

```

```

for(j=0;j<= 999;j++) curtot[j] = 0;
for(j=0;j<= 3999;j++)
{
    curx[j] = 0;
    cury[j] = 0;
}

/* read dig file into arrays for quick display */
numscan = 1;
while (numscan >= 1)
{
    numscan = fscanf(datfil,"%f %f %d",&cury[count],&curx[count],&curtot[cnt]);
    if (numscan >= 1)
    {
        tmp = count + curtot[cnt] - 1;
        count = count + 1;
        for (j=count;j<tmp;j++)
            numscan = fscanf(datfil,"%f %f %d",&cury[j],&curx[j],&dum);
        cnt = cnt + 1;
        count = j;
    }
}
fclose(datfil);
}

void plot_digfils()
{
    char *filnam;
    int nscan,dum,handle;
    int inside,res,found,first;
    int lno,x1,x2,y1,y2;
    float xx1,xx2,yy1,yy2;
    FILE *fil;

    handle = open(digtbl,O_RDONLY);
    fil = fdopen(handle,"r");
    if (fil == NULL)
        printf("Cannot open dig table");
    found = FALSE;
    first = TRUE;
    filnam = (char *) malloc(20);
    strcpy(filnam,refer);
    nscan = fscanf(fil,"%s %d %d %d %d %s",&x1,&x2,&y1,&y2,&lno,&ptfil);
    while (nscan >= 1)
    {
        xx1 = x1;
        xx2 = x2;
        yy1 = y1;
        yy2 = y2;
        inside = FALSE;
        inside = checkranges(&xx1,&xx2,&yy1,&yy2);
        if (lno == levelno)
            res = TRUE;
        else
            res = FALSE;
    }
}

```

```

    if ((inside == TRUE) && (res == TRUE))
    {
        readcurrent();
        plotpts();
        found = TRUE;
        first = FALSE;
    }
    nscan = fscanf(fil,"%s %d %d %d %d %s",&x1,&x2,&y1,&y2,&lno,
    ----->ptfil);
}

if ((found == FALSE) && (first == TRUE) && (strcmp(digtbl,"phys.tbl") != 0))
{
    strcpy(refer,"world3.out");
    readcurrent();
    plotpts();
}

    strcpy(refer,filnam);
    free(filnam);
    fclose(fil);
}

void mainmenu()
{
    strcpy(cont,"global view");
    strcpy(refer,"world3.out");
    strcpy(digtbl,"tile.tbl");
    strcpy(ptfil,"world.pts");
    cxmx = 180;
    cxmn = -180;
    cymx = 90;
    cymn = -90;
}

void setwin()
{
    ox1 = xmn;
    ox2 = xmax;
    oy1 = ymn;
    oy2 = ymx;
}

void def_scr()
{
    int xbar,ybar;

    ybar = (view.bottom - view.top)/2;
    xbar = (view.right - view.left)/3;

    one_xmin = view.left;
    one_xmax = xbar;
    one_ymin = view.top;
    one_ymax = ybar;

    two_xmin = view.left;
    two_xmax = xbar;
    two_ymin = ybar;
    two_ymax = view.bottom;
}

```

```

three_xmin = xbar;
three_xmax = view.right;
three_ymin = view.top;
three_ymax = view.bottom;
)

void setview(int *viewnum)
{
    switch (*viewnum)
    {
        case 1 : {
            setviewport(one_xmin,one_ymin,one_xmax,one_ymax,1);
            break;
        }
        case 2 : {
            setviewport(two_xmin,two_ymin,two_xmax,two_ymax,1);
            break;
        }
        case 3 : {
            setviewport(three_xmin+3,three_ymin+3,three_xmax-3,three_ymax-3,1);
            break;
        }
        case 4 : {
            setviewport(view.left,view.top,view.right,view.bottom,1);
            break;
        }
    }
}

void setup()
{
    setcolor(14);
    setlinestyle(0,5,3);
    rectangle(one_xmin,one_ymin,one_xmax,one_ymax);
    rectangle(two_xmin,two_ymin,two_xmax,two_ymax);
    rectangle(three_xmin,three_ymin,three_xmax,three_ymax);
}

int transformx(float oldx)
{
    int newx;
    newx = ((three_xmax - three_xmin)*(oldx-xmin))/(xmx-xmin);
    return(newx);
}

int transformy(float oldy)
{
    int newy;
    newy = ((three_ymax - three_ymin)*(oldy - ymin))/(ymx - ymin);
    if (newy <= three_ymax)
        newy = abs(newy - three_ymax);
    else
        newy = -5;
    return(newy);
}

void plotpts()
{

```

```

int i,cntr,j;
int startx,starty,nextx,nexty;

i=1;
j=1;
setlinestyle(0,5,1);
setcolor(9);

yratio = (three_ymax - three_ymin)/yrange;
xratio = (three_xmax - three_xmin)/xrange;

while(curtot[i] > 0)
{
    startx = transformx(curx[j]);
    starty = transformy(cury[j]);
    moveto(startx,starty);
    for (cntr=2;cntr<=curtot[i];cntr++)
    {
        nextx = transformx(curx[j+1]);
        nexty = transformy(cury[j+1]);
        lineto(nextx,nexty);
        moveto(nextx,nexty);
        j++;
    }
    j++;
    i++;
}

void setlevel()
{
    xrange = xmax-xmin;
    if (xrange < 0) xrange = -xrange;
    yrange = ymax-ymin;
    if (yrange < 0) yrange = -yrange;
    if ((xrange < 10) || (yrange < 10))
        levelno = 3;
    else
        if ((xrange < 50) || (yrange < 50))
            levelno = 2;
        else
            levelno = 1;
}

void readpts()
{
    FILE *f1;
    float x,y,x1,x2;
    char *place;
    int numscan;

    place = (char *) malloc(25);
    f1 = fopen("ptfil","r");
    if (f1 == NULL)
        printf("cannot open %s\n",ptfil);
    settxtstyle(2,0,1);
    setcolor(2);

```

```

numscan = fscanf(fil,"%s %f %f",place,&y,&x);
while (numscan >= 1)
{
    if ((x >= xmn) && (x <= xmx) && (y >= ymn) && (y <= ymx))
    {
        x2 = transformx(x);
        y2 = transformy(y);
        putpixel(x2,y2,15);
        if (levelno == 3)
        {
            x2 = x2 + 5;
            y2 = y2 + 5;
            if (x2 >= three_xmin)
                outtextxy(x2,y2,place);
        }
        numscan = fscanf(fil,"%s %f %f",place,&y,&x);
    }
    free(place);
    fclose(fil);
}

```

```

float reverse_transformx(int oldx)
{
    float xcoord;

```

```

    xcoord = (oldx)*(xmx - xmn)/(three_xmax - three_xmin) + xmn;
    return(xcoord);
}

```

```

float reverse_transformy(int oldy)
{
    float ycoord;

```

```

    ycoord = (abs(oldy-three_ymax)*(ymx - ymn))/((three_ymax - three_ymin) + ymn;
    return(ycoord);
}

```

```

void text()
{
    char *xmin,*xmax,*ymin,*ymax;

```

```

    xmin = (char *) malloc(7);
    xmax = (char *) malloc(7);
    ymin = (char *) malloc(7);
    ymax = (char *) malloc(7);

    gcvt(xmn,2,xmin);
    gcvt(xmx,2,xmax);
    gcvt(ymn,2,ymin);
    gcvt(ymx,2,ymax);

    setcolor(2);
    settextstyle(2,0,1);
    viewnum = 2;
    setview(&viewnum);

```

```

    outtextxy(10,5,"DISPLAY INFO");
    outtextxy(4,18,"View: ");
    outtextxy(50,18,cont);
}

```

```

    outtextxy(4,30,"Upper Left Corner");
    outtextxy(4,40,"lat: ");
    outtextxy(70,40,"lon: ");
    outtextxy(35,40,ymax);
    outtextxy(100,40,xmin);

    outtextxy(4,55,"Lower Right Corner");
    outtextxy(4,65,"lat: ");
    outtextxy(70,65,"lon: ");
    outtextxy(35,65,ymin);
    outtextxy(100,65,xmax);

```

```

    viewnum = 1;
    setview(&viewnum);
    outtextxy(7,7,"UCSB BROWSE - SELECT AREA");
    outtextxy(25,18,"USER MENU");
    outtextxy(8,28,"1: ZOOM");
    outtextxy(8,38,"2: UNZOOM");
    outtextxy(8,48,"3: GO BACK ONE LEVEL");
    outtextxy(8,58,"4: SELECT CURRENT WINDOW");
    outtextxy(8,70,"PLEASE MAKE A SELECTION");
    outtextxy(30,80,"-->");

```

```

    free(xmin);
    free(xmax);
    free(ymin);
    free(ymax);
}

```

```

void text2(int x, int y)
{
    char *xval,*yval;
    float cx,cy;

```

```

    xval = (char *) malloc(7);
    yval = (char *) malloc(7);

    cx = reverse_transformx(x);
    cy = reverse_transformy(y);

```

```

    gcvt(cx,2,xval);
    gcvt(cy,2,yval);

```

```

    viewnum = 2;
    setview(&viewnum);
    clearviewport();

```

```

    viewnum = 4;
    setview(&viewnum);

```

```

    setcolor(11);
    settextstyle(2,0,3);
    setlinestyle(0,5,3);
    moveto(two_xmin,two_ymin);
    lineto(two_xmax,two_ymin);
    lineto(two_xmax,two_ymax);
    lineto(two_xmin,two_ymax);
    lineto(two_xmin,two_ymin);

```

```

    viewnum = 2;
    setview(&viewnum);

```

```

    outtextxy(10,5,"SELECT WINDOW");
}

```

```

outtextxy(4,30,"n: move cursor north");
outtextxy(4,40,"s: move cursor south");
outtextxy(4,50,"e: move cursor east ");
outtextxy(4,60,"w: move cursor west ");

outtextxy(4,75,"Current coordinates");
outtextxy(4,95,"lat: ");
outtextxy(70,95,"lon: ");

outtextxy(35,95,yval);
outtextxy(100,95,xval);

outtextxy(3,120,"Hit enter to select corner");

viewnum = 3;
setview(&viewnum);
}

void update(int x, int y)
{
    char *xval,*yval;
    float cx,cy;
    int fillarea[] = {3,90, 135,90, 135,105, 3,105, 3,90 };

    xval = (char *) malloc(7);
    yval = (char *) malloc(7);

    cx = reverse_transform(x);
    cy = reverse_transform(y);

    gcvtx(cx,2,xval);
    gcvty(cy,2,yval);

    viewnum = 2;
    setview(&viewnum);

    setcolor(0);
    fillpoly(5,fillarea);
    setcolor(9);

    outtextxy(4,95,"lat: ");
    outtextxy(70,95,"lon: ");
    outtextxy(35,95,yval);
    outtextxy(100,95,xval);

    viewnum = 3;
    setview(&viewnum);
}

void choosewin()
{
    int x,y;
    int ch;

    /* declarations for cursor */
    int triangle[] = {10,10, 20,13, 10,15, 10,10};
    int ulx, uly, lrx, lry;
    unsigned size;

```

```

/* Draw cursor */
setfillstyle(SOLID_FILL, getmaxcolor());
fillpoly(sizeof(triangle)/(2*sizeof(int)), triangle);

ulx = 10;
uly = 10;
lrx = 20;
lry = 15;

size = imagesize(ulx,uly,lrx,lry);
cursor = (void *) malloc( size );
getimage(ulx,uly,lrx,lry, cursor);

x = ulx;
y = uly;

/* get first point */
text2(x,y);
ch = 0;
while (ch != 13)
{
    ch = getch();
    putimage(x, y, cursor, XOR_PUT);
    if (ch == 110) y=y-4; /* north */
    if (ch == 115) y=y+4;
    if (ch == 101) x=x+6; /* east */
    if (ch == 119) x=x-6;
    putimage(x, y, cursor, XOR_PUT);
    update(x,y);
}

x1 = reverse_transform(x+10);
y1 = reverse_transform(y+3);

outtextxy(x,y,"+");
putimage(x,y,cursor,XOR_PUT);

/* get second point */
ch = 0;
while (ch != 13)
{
    ch = getch();
    putimage(x, y, cursor, XOR_PUT);
    if (ch == 110) y=y-6;
    if (ch == 115) y=y+6;
    if (ch == 101) x=x+9;
    if (ch == 119) x=x-9;
    putimage(x, y, cursor, XOR_PUT);
    update(x,y);
}

x2 = reverse_transform(x+10);
y2 = reverse_transform(y+2);

free( cursor );
}

void zoom()
{
    float deltax,deltay,tmpx,tmpy;

```

```

float ave, range;
deltax = xmx-xmn;
deltay = ymx-ymn;
if (x1 > x2)
{
    tmpx = x1;
    x1 = x2;
    x2 = tmpx;
}
if (y1 > y2)
{
    tmpy = y1;
    y1 = y2;
    y2 = tmpy;
}
tmpx = (x2-x1)/deltax;
tmpy = (y2-y1)/deltay;
if (tmpx > tmpy)
{
    ave = (y1 + y2)/2;
    range = deltax/2 * tmpx;
    y1 = ave - range;
    y2 = ave + range;
}
else
{
    ave = (x1 + x2)/2;
    range = deltax/2 * tmpy;
    x1 = ave - range;
    x2 = ave + range;
}

void setbounds()
{
    xmn = x1;
    xmx = x2;
    ymn = y1;
    ymx = y2;
}

void record_window()
{
    int n;
    curlevel = curlevel + 1;
    n = 2*curlevel - 1;
    xrec[n] = xmn;
    xrec[n+1] = xmx;
    yrec[n] = ymn;
    yrec[n+1] = ymx;
}

void unzoom()
{
    xmn = cxmn;
    xmx = cxmx;
    ymn = cymn;

```

```

    ymx = cymx;
}

void goback()
{
    int n;
    curlevel = curlevel - 1;
    n = 2*curlevel - 1;
    if (curlevel <= 0)
    {
        viewnum = 4;
        setview(&viewnum);
        clearviewport();
        zoomed = FALSE;
    }
    else
    {
        xmn = xrec[n];
        xmx = xrec[n+1];
        ymn = yrec[n];
        ymx = yrec[n+1];
    }
}

int checkranges(float *xx1,float *xx2,float *yy1,float *yy2)
{
    int flag;
    flag = 0;
    if ((xmn < *xx2) && (xmn > *xx1) && (ymx < *yy2) && (ymx > *yy1))
        flag = 1;
    else if ((xmx > *xx1) && (xmx < *xx2) && (ymx > *yy1) && (ymx < *yy2))
        flag = 1;
    else if ((xmx > *xx1) && (xmx < *xx2) && (ymn < *yy2) && (ymn > *yy1))
        flag = 1;
    else if ((xmn > *xx1) && (xmn < *xx2) && (ymn > *yy1) && (ymn < *yy2))
        flag = 1;
    /* check to see if dig file completely contained in zoom window */
    else if ((xmn < *xx1) && (xmx > *xx2) && (ymn < *yy1) && (ymx > *yy2))
        flag = 1;
    /* check to see if dig file intersects zoom window */
    else if ((*xx1 > xmn) && (*xx1 < xmx) && (*yy2 > ymn) && (*yy2 < ymx))
        flag = 1;
    else if ((*xx2 > xmn) && (*xx2 < xmx) && (*yy2 > ymn) && (*yy2 < ymx))
        flag = 1;
    else if ((*xx2 > xmn) && (*xx2 < xmx) && (*yy1 > ymn) && (*yy1 < ymx))
        flag = 1;
    else if ((*xx1 > xmn) && (*xx1 < xmx) && (*yy1 > ymn) && (*yy1 < ymx))
        flag = 1;
    else
        flag = 0;
}

```

return(flag);

}

Meeting Attendees

The following is the list of attendees at the team meetings we have sponsored during the first contract year.

Frequent irregular meetings through the year at UCSB, for local faculty:

David Simonett
Raymond Smith
Earl Hajic
Frank Davis

Spring meeting at UCSB - May 1987

Dr. Ron Blom, JPL
Mr. Jim Brass, ARC
Dr. Supriya Chakrabarti - Space Sciences Lab, UC Berkeley
Mr. Len Gaydos, USGS Menlo Park
Dr. Thomas Logan - JPL
Prof. Doug Stowe - Geography, San Diego State Univ.

Spring meeting at GSFC - June 1987

Prof. Peter Cornillon - Oceanography, Univ. Rhode Island
Prof. Bob Ragan - Civil Engineering, Univ. Maryland
Dr. Nick Faust - Georgia Tech Research Inst., Atlanta
Dr. Forest Hall - GSFC
Dr. Paul Smith - GSFC (now at NASA Headquarters)

Winter meeting at UCSB - December 1987

Dr. Bob Chase - Univ. Colorado, Boulder
Dr. Dick Collier - Purdue Univ., NPIRS
Mr. Ralph Dedecker - SSEC, U. Wisconsin, Madison
Prof. John Jensen - Geography, Univ. South Carolina
Dr. Tom George - Geophysical Institute, U. Alaska, Fairbanks
Dr. Nick Faust - Georgia Tech Research Inst.

International Meetings

During the past year, the University of California at Santa Barbara (UCSB) Information Sciences Research Group (ISRG) personnel have become increasingly involved in activities associated with the United Nations Environment Programs (UNEP) Global Environmental Monitoring System (GEMS) Global Resource Information Database (GRID) activity.

During September 1987, Dr. Star participated in a meeting held at the United Nations Environment Programme offices in Nairobi, Kenya. Particular emphasis at this meeting was on the information systems support that the GRID program will require as it becomes operational over the next several years. The invitation to join this working group came from Dr. Harvey Croze, GRID coordinator.

At this meeting, the working group reviewed the history, status, and future objectives of the GRID program. The pilot program, which began in 1984, was based on software and hardware donated or loaned to UNEP by government and private organizations in Europe and the United States. After two years of the pilot program, a decision was made by the UNEP in the summer of 1987 to begin an implementation phase for GRID. This next phase is to begin in early 1988 and continue for two years. Our meeting in September 1987 came out of this decision; we were to provide guidance for the start of this next phase.

A second element of our discussions in September regarded long-term data management issues. At this time, there is a relatively small volume of data in the GRID system. Plans must be made to begin to develop ways to manage the future large data volumes, as well as provide means for users to access both the data itself as well as descriptive information about the data. I made a slide presentation to both the GRID staff and the working group on the BROWSE testbed activity, funded through NASA Headquarters. The BROWSE testbed, at a conceptual level, provides input to their planning in several ways. First, it demonstrates a way to index spatial data holdings in terms of prescribed attributes (geographic coverage, date of compilation, thematic information, etc.). These attributes are then freely searched by the users to identify relevant datasets. Second, since the BROWSE testbed includes capabilities to store limited image datasets, the same system provides limited access to the data itself. This is particularly useful in the case of the UNEP/GRID program, since there are, at present, no methods for charging users fees even to offset database duplication costs. A future system similar to BROWSE could provide electronic access to both data and data descriptions with no direct staff costs. Croze and Mooneyhan were extremely interested in our development. I have forwarded copies of the BROWSE user guide to them, and Dr. Croze has scheduled a visit to UCSB to see the testbed in action and have further discussions with us.

A related meeting was held in Kew, England, at the World Conservation Monitoring Centre. At this meeting, funded by the United Nations, the overall architecture of WCMC's data and information system was evaluated, as a starting point for hardware and software system migration in the future. The BROWSE testbed was again a focus of discussion, as a prototype user interface for non-expert data system users. WCWM is

Browse in the EOS Era

funded by the United Nations Environment Programme, International Union for the Conservation of Nature, and the World Wildlife Fund, and representatives of these agencies attended.

Based on this first meeting, we have submitted a letter of interest to the United Nations Environment Programme, to begin to develop some of the elements of a distributed data and information system for WCMC. UNEP staff indicated that they are interested in locating funds for our participation in this effort from their sources, and we look forward to this new collaboration with UNEP and WCMC.

Recent Testbed Users

The following records indicate some of the recent users of our system, based on their first date of use. These are in addition to our team of collaborators.

Date: 7-OCT-88 Current Pacific Coast Time: 14:52:55
User: J.B.Dalton
Institution: NASA/Ames Research Center
Discipline: Planetary Atmospheres, Geophysics
Hardware: HDS 3200 model 30 (VT220,TEK40) Access: DIAL-UP/MODEM

Date: 17-OCT-88 Current Pacific Coast Time: 12:02:58
User: Matt Smith
Institution: Marshall Space Flight Center
Discipline: Atmospheric Science
Hardware: IBM PS/2 80 Access: SPAN/DECNET

Date: 17-OCT-88 Current Pacific Coast Time: 12:43:04
User: Fred Waltz
Institution: Eros Data Center
Hardware: modgraph Access: SPAN/DECNET

Date: 18-OCT-88 Current Pacific Coast Time: 12:28:36
User: Dr. Tom George
Institution: University of Alaska
Discipline: Interdisciplinary
Hardware: MAC II via SPAN Access: SPAN/DECNET

Date: 26-OCT-88 Current Pacific Coast Time: 13:39:42
User: Dr. Chester Richmond
Institution: Oak Ridge National Lab
Discipline: biology
Hardware: microvax Access: SPAN/DECNET

Date: 27-OCT-88 Current Pacific Coast Time: 09:50:48
User: Prof. R. Chase
Institution: Colorado University, Boulder
Discipline: large scale hydrodynamics
Hardware: sun Access: ARPANET/NSFNET

Date: 4-NOV-88 Current Pacific Coast Time: 10:09:36
User: Michael Goodman
Institution: Marshall Space Flight Center
Discipline: none
Hardware: pc/at Access: DIAL-UP/MODEM

Date: 8-NOV-88 Current Pacific Coast Time: 09:12:40

Information Sciences Research Group
University of California, Santa Barbara

Browse in the EOS Era

User: Prof. Ray Arvidson
Institution: Washington University, St. Louis
Discipline: geoscience
Hardware: vax750 Access: SPAN/DECNET

Date: 10-NOV-88 Current Pacific Coast Time: 14:40:01
User: Joe Bredekamp
Institution: Nasa Headquarters
Discipline: info sys
Hardware: vax Access: SPAN/DECNET

Date: 14-NOV-88 Current Pacific Coast Time: 15:53:15
User: Dr. Heiner Biesel
Institution: Evans and Sutherland
Discipline: 1
Hardware: vaxstation Access: SPAN/DECNET

Date: 16-NOV-88 Current Pacific Coast Time: 15:47:56
User: Condor Recovery team
Institution: California Department of Fish and Game, USFWS, Audubon
Discipline: Ecology
Hardware: microvax Access: SPAN/DECNET

Date: 12-JAN-89 Current Pacific Coast Time: 14:16:36
User: David Salisbury
Institution: UCSB
Discipline: public relations
Hardware: vax Access: SPAN/DECNET

Date: 19-JAN-89 Current Pacific Coast Time: 14:17:41
User: Dr. John Good
Institution: California Institute of Technology/IPAC
Discipline: Astrophysics
Hardware: pc/color/1200 baud modem Access: DIAL-UP/MODEM

Date: 19-JAN-89 Current Pacific Coast Time: 16:23:45
User: Tadas Sesplaukis
Institution: California Institute of Technology/IPAC
Discipline: astronomy
Hardware: ibm/pc Access: ARPANET/NSFNET

Date: 16-FEB-89 Current Pacific Coast Time: 10:39:32
User: Peter Price
Institution: Marathon Oil Company, Houston
Discipline: Remote Sensing
Hardware: AMIGA(IBM-PC COM) Access: DIAL-UP/MODEM

Information Sciences Research Group
University of California, Santa Barbara

Browse in the EOS Era

Date: 23-FEB-89 Current Pacific Coast Time: 12:28:41
User: CHarles Martin
Institution: Lockheed
Discipline: Image Processing
Hardware: MICROVAX II Access: SPAN/DECNET

Date: 2-MAR-89 Current Pacific Coast Time: 15:11:17
TEK 4010 terminal
User: Dr. Harry Miles
Institution: The Nature Conservancy
Discipline: yes
Hardware: vaxstation Access: SPAN/DECNET

Date: 4-MAR-89 Current Pacific Coast Time: 12:16:20
User: Hunter Poole
Institution: Lockheed
Discipline: remote sensing analysis
Hardware: MICROVAX II Access: SPAN/DECNET

Date: 7-MAR-89 Current Pacific Coast Time: 10:44:51
User: Mr. T.M. Albert
Institution: USGS Information Systems
Discipline: none
Hardware: vaxstation Access: SPAN/DECNET

Date: 12-MAR-89 Current Pacific Coast Time: 13:37:36
User: Fayne Sisco
Institution: IBM Houston, Federal Systems Division
Discipline: System Engineering
Hardware: PS/2 Mod 70 Access: DIAL-UP/MODEM

Date: 27-APR-89 Current Pacific Coast Time: 14:00:22
User: Lance Goode
Institution: Navigation Technologies
Discipline: none
Hardware: vax station Access: SPAN/DECNET

BROWSE User's Guide

Build 2

NASA BROWSE Testbed Facility

Principal Investigators:

**Dr. John E. Estes
Dr. Jeffrey L. Star**

Co-investigators:

**David M. Stoms
Mark A. Friedl**

Draft--09/01/88

Research conducted under NASA Grant NAGW-987

**University of California, Santa Barbara
Remote Sensing Research Unit**

Table of Contents

1. Introduction
 - 1.1. Major Revisions of Build 2
 - 1.2. Purpose of Document
 - 1.3. Scope of Document
 - 1.4. Definitions
2. BROWSE Concept
 - 2.1. Definition of BROWSE Capability
 - 2.2. Need for Browsing
 - 2.3. Pilot Data Systems and EosDIS
3. BROWSE Subsystems -- General Explanation
 - 3.1. Database Management System and Databases
 - 3.2. User Interface System
 - 3.3. Image Display System
 - 3.4. Communications System
 - 3.5. Image Processing System
 - 3.6. BROWSE Facility Hardware
 - 3.7. User Hardware Requirements
4. Detailed Description of the Use of the BROWSE Testbed
 - 4.1. Logging on to BROWSE
 - 4.2. User Interface
 - 4.2.1. Using the INVENTORY Database
 - 4.2.2. Contents of the INVENTORY Database
 - 4.2.3. Using the DIRECTORY Database
 - 4.3. Displaying BROWSE Images
5. References
6. Appendices--available on request
 - 6.1 IDS Documentation
 - 6.2 Database Schemas

BROWSE User's Guide

Build 2

1. Introduction

1.1. Major Revisions in Build 2

Based on feedback from BROWSE users, the following revisions have been made in the BROWSE software for this new version:

1) The electronic map has been improved to make it display much faster. Also, for selected areas, as you zoom in, the map detail increases.

2) Semi-automatic procedures have been added for transferring browse files with DECNET and TCP/IP protocols, in addition to KERMIT for dial-up mode.

3) The term 'CATALOG' has been changed to 'INVENTORY' throughout the BROWSE testbed to conform with the new ESADS and DIF lexicons.

4) Traps for erroneous user input have been added so the user won't be dropped from the program.

5) A Quit option has been added to the routine for displaying records from the INVENTORY database, so you don't have to read long lists of records.

6) Prompts and other screen text have been clarified.

7) More browse files from additional sensors have been added to the testbed.

1.2. Purpose of Document

This document updates the details on the use of the NASA BROWSE testbed facility at the University of California, Santa Barbara. Basic and applied research conducted on this project were made under the auspices of NASA Grant NAGW-987. BROWSE is designed to investigate the concepts that would allow scientists to locate image and other spatial data in distributed archives and

to preview that data from terminals in their offices. As such, BROWSE is a prototype and not a fully polished, operational information system.

1.3. Scope of Document

This document contains a description of BROWSE from the user's standpoint. The underlying concepts of BROWSE are briefly explained in chapter 2, followed by an overview of the system components in chapter 3. Chapter 4 presents details on the use of each BROWSE component, including menus and example queries.

1.4. Definitions

Attribute: description of the column entries in a relation.

Browse file: preprocessed, low resolution file of spatial data accessible through the INVENTORY database.

Browsing: the process of locating and viewing spatial data from a remote terminal.

Catalog interoperability: a capability of permitting a user to access different databases with different schemas and query languages through the use of only one of them.

Database: an integrated collection of occurrences of a number of record types, where the record types and their occurrences are interrelated by various specific relationships.

Directory: top-level index containing information about location, ownership, contents of databases.

Image compression: a method of encoding digital image data with less redundancy to achieve greater efficiency of storage and transmission.

Inventory: descriptions of a database in sufficient detail to retrieve subsets of data. Searchable by data fields or attributes, down to some level of granularity.

Node: a computer facility connected by network to the BROWSE testbed.

Record: a row of data in a relation (i.e., a tuple).

Relation: a table of data in a database.

Relational database: a database developed on the relational model; i.e.,

based on tables.

Schema: a description of the database that is compiled or translated into machine-readable form.

Tuple: a row of data in a relation (i.e., a record).

2. BROWSE Concept

2.1. Definition of BROWSE Capability

The purpose of the NASA grant to UCSB was to examine problems related to browsing of image data sets by scientists from many disciplines at widely distributed locations. One of the problems identified was to define exactly what is meant by "browsing". Generally, "browse" is being used to describe a process of locating image data and pertinent information about that data, as well as obtaining a quick-look (in some form) for a decision concerning acquisition from the host site. The quick-look might involve a compressed display (i.e., reduced spatial, spectral, or radiometric resolution) of the imagery, image histograms, or image statistics. The intent is not to make the full dataset directly available to users nor to duplicate image analysis capabilities.

2.2. Need for Browsing

BROWSE anticipates the high data volumes associated with Eos, the Earth Observing System on the Space Station Complex. The ability to browse Eos data and to select only that portion of the data relevant to a given need will be important.

Others have noted the utility of browsing capability in the conduct of scientific research. The prestigious Committee on Data Management and Computation (CODMAC) of the National Academy of Sciences, in its goals and

priorities to maximize scientific return on our nation's investment in space science recommended more emphasis be given to development of "electronic browsing capability, allowing users to explore data files via communications links". Consequently, they recommended that browsing use quick-look low resolution data (CODMAC, 1982; CODMAC, 1986). The issues concerning a browsing function, as well as descriptions of the BROWSE testbed, are presented in several conference papers written the RSRU staff (Star et al., 1987; Star et al., in press; Stoms et al., 1988).

2.3. Pilot Data Systems and EosDIS

Guidelines for the BROWSE testbed specified that it should consider multidisciplinary needs of the scientific community. Therefore, BROWSE is a prototype module that could be incorporated into any of the NASA pilot data systems for land, climate, oceans, and planets (PLDS, PCDS, NODS, and PDS) and into the Eos Data and Information System. The larger issues of data management in these information systems is being addressed by the pilot studies. BROWSE deals with the quick-look aspect that is common to all.

3. BROWSE Subsystems -- General Explanation

This section gives an overview of the subsystems constituting the BROWSE facility. In terms of physical facilities, BROWSE consists of three components --a host facility at UCSB, the user's remote terminal, and the communications link between them. At the host facility resides the user interface, the database management system (DBMS) with its databases, and a graphics program for selecting geographic area. The user's terminal contains the image display and the communications software. An off-line image processing system is used to preprocess browse images. A project objective was for rapid prototyping so

existing public domain software was used where possible. Details about the operation of each component is described in chapter 4.

3.1. Database Management System and Databases

The relational database of data about available imagery is maintained at the host facility (in the future, each node will maintain a database of its own holdings). We selected RIM (Relational Information Management-JPL Version 4.5) for the DBMS. The NASA Ocean Data System at the Jet Propulsion Laboratory also uses RIM, which originated as a prototype at Boeing Commercial Airplane Company under authorization of NASA contract NAS1-14700. JPL added certain enhancements such as a multiuser capability and Fortran interface subroutines for relational algebra. RIM can be executed in standalone mode from a VAX/VMS prompt and has its own on-line help. For details on RIM, see "Relational Information Management (RIM) User's Guide--715-604" (Jet Propulsion Laboratory, 1985). BROWSE saves the user from having to learn RIM's command language or the details of the database structure by providing a menu-driven front-end, coded in a Fortran applications program. BROWSE has two databases--one about specific images at the host node (INVENTORY) and another about data collections at specific nodes (DIRECTORY).

3.2. User Interface System

The user interface is a set of Fortran subroutines that prompt the user for the desired attributes of the imagery and formulate the appropriate database queries. This makes it easier for the user unfamiliar with either RIM or the specific database structure. The drawback of such an approach is that the queries are restricted to predetermined primary attributes. Consequently, flexibility is traded-off for ease of use.

3.3. Image Display System

This software module, named IDS for Image Display System, is loaded into the user's terminal from floppy disk. Currently this software is written only for IBM PC-AT with 640 kbytes of memory. Using a windowing environment, it displays the browse image that has been transmitted from the host to the terminal. Statistics and a histogram are simultaneously displayed with the image. Limited image processing functions, basically contrast stretching, are provided. The intent is on display, not image analysis. Therefore, only preprocessed imagery can be browsed. At this point, IDS is programmed to interpret headers from ERDAS files. ERDAS is a commercial image processing system. Any image display software available at the user's local worksite can be substituted for IDS, if desired.

3.4. Communications System

Access to the BROWSE system is available over a regular telephone line or network access via SPAN/DECNET and ARPANET/NSFNET. Transfer of browse files is provided using these network protocols. For dial-up, terminal emulation software, such as VTERM and KERMIT, can be used to access the host node. Both of these software packages provide both the KERMIT protocol for transferring browse files and a TEKTRONIX emulator for the graphics program for selecting geographic area.

3.5. Image Processing System

The browseable image data on-line at the host facility has been preprocessed locally, using a commercial PC-based image processing system. Imagery has been subset from large data files, processed and often compressed, and transferred on magnetic tape to the host minicomputer.

3.6. BROWSE Facility Hardware

Standard hardware has been selected for the testbed facility. The host facility at UCSB consists of a MicroVAX II workstation with approximately 700 megaBytes of disk memory, a dual-speed 9-track tape drive, and a VT260 terminal. Currently, we are using a Telebit Trailblazer Plus modem with up to 9600 baud capability for dial-up access to the testbed. The computer is networked to an IBM PC-AT.

3.7. Hardware Requirements

The BROWSE testbed was designed to accommodate standard commercial hardware. For the user only interested in querying the database without using the electronic map or IDS, a PC with VT100 emulating software and a modem is adequate. To use the graphics and image display packages with an IBM-PC requires the addition of a Tek 4010/4014 emulator, cursor control, an EGA (Enhanced Graphics Adapter) board, and at least 640 Kbytes of RAM. Any terminal with Tektroniks 4010 capability can use the graphic mode of BROWSE (i.e., the electronic map).

4. Detailed Description of the Use of the BROWSE Testbed

This section describes the operation of the BROWSE components outlined above. It is written directly to the user in terms of procedures, prompts to expect, and general navigation through the system.

4.1. Logging On to BROWSE

There are three methods to connect to the BROWSE facility:

1. From your local personal computer with a modem, dial 805-961-8409 using

VTERM. You will need a terminal emulator for VT100 and/or TEK4010. Our modem has 9600 (PEP protocol), 2400, 1200, and 300 baud capability.

2. For ARPANET or NSFNET users, type **TELNETSBRSRU.UCSB.EDU**.

3. For SPAN/DECNET users, type **SETHOST 45177**.

At the appropriate point, the system will respond with:

SERSRU - UCSB Remote Sensing Research Unit VAXstation II/VMS 4.5

Username:
Password:

To these prompts, you should enter **GUEST** and **J_DODGE** respectively. Note that this password lets you query the database but not to do any irreparable damage to it or any other files on the computer disks. If someone else is already running **BROWSE**, you will see a message to this effect on your terminal. Please try again later. Normally, though, you will see some logon greetings. The Browse program runs automatically. First you will be asked what terminal type you are using. This is necessary for the program to provide appropriate control codes such as for clearing the screen. Next you will see the **BROWSE** welcome banner. The bottom line of the banner lists the current number of records in the **INVENTORY** database and the number of on-line browse files. Just hit a carriage return at the prompt to continue. Then you will be asked for your name, institution or agency, your professional discipline, computer type, and connection mode for our records. The connection mode is also used to select the appropriate protocol for browse datafile transfer.

4.2 User Interface

The user interface consists of a set of menus and prompts to guide you through the process of querying either of the two databases. From menus, enter the number of the option you want. Other prompts are relatively self-

explanatory. The main menu, shown in Figure 1, lets you choose to query the databases (the usual option). QUIT logs you out of the BROWSE testbed. HELP presents a menu of terms to choose from and then the associated definition for your choice. The HELP function is merely a prototype at this stage and only provides definitions of terminology. The MESSAGES option provides a forum for user feedback as well as update notes from the system developers.

```
UCSB-BROWSE
Build 2      Revised 09/01/88

MAIN MENU

1. QUERY DATABASES
2. BROWSE IMAGES
3. HELP
4. COMMENTS & MESSAGES
5. QUIT BROWSE

ENTER CHOICE NUMBER:
```

Figure 1. Main Menu

Normally, you will pick option 1, selected by typing the number and a carriage return. The next menu, QUERY, lets you select which database you want to consult. INVENTORY contains data about specific images and DIRECTORY generally describes the holdings at BROWSE nodes, whether on-line or off. The USER database would contain information about applications scientists who may be able to provide advice or collaboration on specific disciplines, sensor data,

or geographic regions. For the rapid prototype, the USER database is not implemented.

4.2.1 Using the INVENTORY Database

The INVENTORY menu presents the choice of attributes that you may use to restrict a query. While this limits your flexibility in manipulating data, hopefully the gain in ease of querying on the most commonly used attributes is a reasonable trade-off. These attributes include geographic area, platform/sensor, date of acquisition, cloud cover, and on-line status. Other options allow you to view the current values of query attributes (including defaults if unchanged) and the summary of the current set of records retrieved from the INVENTORY, or if desired, the complete record of each image retrieved. You can also save these records in a disk file for later viewing or printing.

Warning In this version of BROWSE, you should only make one query based on each attribute. Each new query projects from the previous one. So if you try, for instance, to query on platform name "Landsat", followed by a query on platform name "Aircraft", the intersection will be an empty set. On the other hand, queries that narrow a previous search, such as a shorter date range, will give the expected results.

Prompts for the values of query attributes are essentially self-explanatory. For platform name, you are presented a list of platforms. You simply enter the number associated with your choice. The "All" choice makes no reduction in the current set of records retrieved. Once you have chosen a platform, the choices for sensor are limited to those available on that platform, preventing you from trying to pick an AVHRR sensor on Landsat. (In the future, the sensor attribute may be modified so that you can specify a sensor type, e.g., "Microwave" as well as the specific name such as "SIR-A").

The geographic area of interest can be specified in either graphic or keyboard mode, depending on whether you select TEK or VT100 mode at the beginning of the session. Graphic mode lets you use a mouse (or cursor arrow keys) to select a bounding rectangle on an electronic map display. First, you must select an option from the menu in the upper left corner of the display window by positioning the cursor on your choice and hitting the <RETURN> key. A ">" will indicate your selection. Usually, your first choice will be Zoom, to pick a specific region. Select the northwest and southeast corners of an area by moving the cursor to the desired location and press the <RETURN> key on the keyboard. The map will be redrawn, zoomed to the area you have delineated. This can be repeated as often as you like, and you can **unzoom** back to the previous map or to the globe if you are unsatisfied with the area drawn. As you zoom, increasing detail is added to the map (at least for selected regions to demonstrate the concept). See Figure 2 for an example of the electronic map. When you have the area you want, select option 4 from the menu which queries the INVENTORY based on your selected coordinates. The keyboard method in VT100 mode prompts you to type in the northwest and southeast coordinates from the keyboard in the exact format described on the terminal.

For specifying acquisition dates, you have several options. You may select a start and end date of a range, a specific date, or a specific month, for instance, if you wanted June scenes of an area but the year was unimportant.

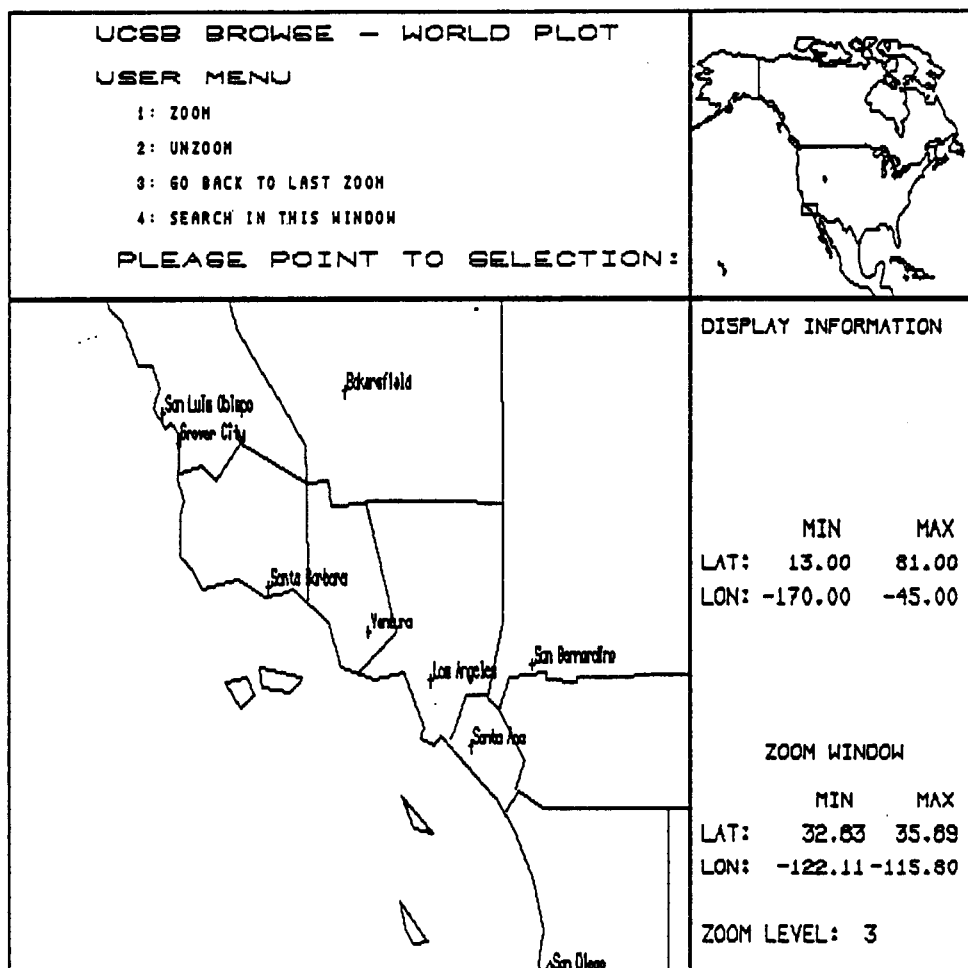


Figure 2. Example of Electronic Map Selecting a Region around Santa Barbara, California

The cloud cover attribute simply sets a maximum limit of the percentage of cover you will tolerate. Some scenes have an unknown cloud cover and are coded with a value of -1 in the INVENTORY database. Therefore, they will be included in all queries restricted by cloud cover but the user is cautioned about their significance.

When you have finished specifying attributes, or after any single query, you can examine the selected records from the INVENTORY by selecting option 7 from the menu. A summary of each record is displayed, one screenful at a time, for ease of reading. You have the option of sorting the output by several attributes before displaying the records. You can also type Q to leave the display and return to the INVENTORY menu. At the end, you are asked if you want to see details of any particular record (an example of which is shown in Figure 3). If you do, enter the number of the desired record (not the scene ID number but the number on the far left of each row). You will also be given the opportunity to save that record in a disk file for later viewing or printing. This may be helpful for remembering the image file names or attributes. You can select as many records as you like to be saved.

```

7) SCENE ID: YS020010043X0      DATE: 12/14/84
   PHYSICAL DESC: SO.CAL COAST    PLATFORM: LANDSAT
   HISTORY: NIR/VIS.SUBSET        MISSION: 5
   QUALITY: NULL                  SENSOR: TM
   FILE POINTER: C.TM1SANTAB0044R3.RAT  CLOUD COVER: 0%
   FORMAT: DIGITAL                ON-LINE? ON

   CENTER LATITUDE : +34.599998      X-GRID: 42
   CENTER LONGITUDE: -119.816666      Y-GRID: 36
   PLACE NAME: Santa Barbara, CA

UCSB-BROWSE

Would You like to save this record in a disk file? (Y/N): n
Using the standard VAX DCL COPY command
At the To: , type Your address/path/file in format
node"username password": device:[path]file.ext

Directory $DISK3:[BROWSE.TM]
SANTAB0044R3.RAT:1      1545 15-SEP-1987 10:23
Total of 1 file, 1545 blocks.

Would You like to transfer this Browse file to Your terminal? (Y/N):

```

Figure 3. Example Results of a Query for Data
of Santa Barbara, California

After looking at a detailed record of an on-line browse file, you will be asked if you want to transfer it to your terminal. If you do, BROWSE invokes the appropriate file transfer protocol for your type of connection (KERMIT for dial-up users, COPY for DECNET or SPAN, and FTP for the TCP/IP networks). For

dial-up mode, BROWSE will automatically invoke KERMIT and pass the filename found in the INVENTORY record. For a user with VTERM on a PC compatible, you must press the ALT and 'K' keys simultaneously on your PC keyboard, when you see characters like ,Sp/@-#Y1~Y. At the VT-KERMIT> prompt, type receive. The transfer at 9600 baud may take 5 to 15 minutes, depending on the file size. When the transfer is complete, the PC will beep at you and give you another KERMIT prompt. Type connect to return to BROWSE.

Alternatively, for SPAN/DECNET users, the DCL COPY command is invoked with the browse filename. VMS will prompt you with a To: at which point you should enter a complete path/filename where you want the file to be placed. Use the format: node"logname password"::device:[directory]filename . For TCP/IP users on ARPANET or NSFNET, you will be prompted for several items. At the FTP AWAITING HOST> prompt, type SET HOST computer name. At the FTP> prompt that results, type LOGIN username, and then your password at your home computer system when requested. Note that this transaction is occurring on the system level so your password is not being intercepted by the BROWSE system. Then at the FTP> prompt type SEND/TYPE=IMAGE filename, using the browse filename. When the FTP> prompt returns, type EXIT to continue. Please let us know if you have problems with these. Leave a comment in the BROWSE Messages utility or send mail to star@sbrsru.ucsb.edu or NASAMAIL JSTAR.

After reviewing the records from the INVENTORY and transferring browse files to your terminal, select option 8 on the INVENTORY menu to return to the Mainmenu.

4.2.2. Contents of the INVENTORY Database

For the testbed, the INVENTORY contains about 600 records for digital imagery archived at the UCSB Remote Sensing Research Unit. A sample of these

scenes are in browseable form. Additional scenes of AVHRR sea surface temperature data for the North Atlantic were provided by Peter Cornillon at the University of Rhode Island.

For each scene, the database describes attributes for scene ID number, center coordinates, cloud cover, on-line status, processing history, image quality, and the full path/filename of the image file. The full database schema is available on request.

4.2.3. Using the DIRECTORY Database

The DIRECTORY database is similar in operation to the INVENTORY database. However, the DIRECTORY describes collections of data, emphasis of the collection, and the address of the facility. Query options in the DIRECTORY menu include platform/sensor combination, geographic area of emphasis, and data format. Conversely, a user can specify a specific node to see what is there. As in the INVENTORY, you can display query attributes and the results of the query. At this point, only the UCSB Remote Sensing Research Unit is on-line in our network. Therefore, accessing other facilities must be done externally to BROWSE.

4.3. Displaying BROWSE Images

When you have finished querying the databases and are ready to look at the images you have selected, choose option 5 on the main menu to leave BROWSE. You will automatically be logged out. However, on a PC using VTERM, you need to type **control/break** to return to DOS. Then run your local image display software. For IBM PC-AT users, we can provide a copy of IDS, a public domain image processing system. It has capabilities for displaying image data with

eight colors, plus a histogram, and statistics. All the commands are from pop-up menus, including options for contrast enhancement. Details of IDS operations are available on request. Of course, you can display the image on your own in-house image processing system, provided you can read files with an ERDAS header. We can provide assistance in changing file formats; contact us at the address on the cover.

5. References

- Committee on Data Management and Computation (CODMAC), 1982.
"Data Management and Computation Volume I: Issues and Recommendations", National Academy Press, Washington, D. C., 167 pp.
- Committee on Data Management and Computation (CODMAC), 1986.
"Issues and Recommendations Associated with Distributed Computation and Data Management Systems for the Space Sciences", National Academy Press, Washington, D. C., 111 pp.
- Jet Propulsion Laboratory, 1985.
"Relational Information Management (RIM) User's Guide--JPL Version 4.5", Pasadena, California, 81 pp.
- Star, Jeffrey L., Mark A. Friedl, David M. Stoms, and John E. Estes, 1987.
"Browse Capability for Spatial Databases", Proceedings of GIS '87 Conference, San Francisco, CA, October 26-29, 1987, pp. 196-205.
- Star, Jeffrey L., David M. Stoms, Mark A. Friedl, and John E. Estes, in press.
"Electronic Browsing for Suitable GIS Data", Proceedings of IGIS Conference, Crystal City, VA, November 15-18, 1987.
- Stoms, David M., Jeffrey L. Star, and John E. Estes, 1988.
"Knowledge-Based Image Data Management: An Expert Front-End for the BROWSE Facility", Technical Papers of the 1988 ACSM-ASPRS Annual Convention, St. Louis, MO, March 13-19, 1988.

BROWSE
Image Display System
Users Manual

Mark A. Friedl

Geography Remote Sensing Unit
University of California, Santa Barbara

May 1989

Image Display System

The BROWSE Image Display System (IDS) is an interactive software package that has been designed to be used for quick viewing of image data on a IBM PC type computer. The system includes a set of disk file I/O operations, as well as image display and enhancement routines which allow the user to view and assess the quality of image data to in order to better assess it's suitability for a specific application. IDS is configured in a menu driven format for easy use, and extensive error checking is included to aid the user in avoiding inadvertent errors.

TABLE OF CONTENTS

I	The IDS System	4
	Introduction	
	System Description	
	Hardware	
	Software	
II	Statistical functions	6
III	Enhancement functions	7
IV	Spatial Filters	8
V	Command Summary	9

Appendicies

A	System Installation	14
---	---------------------	----

System Description

Hardware

The hardware configurations supported by the system include most standard IBM PC devices, except that only "high-resolution" bit-mapped displays are supported, i.e. where the vertical resolution on the screen is at least 350 pixels. Input devices supported include several types of mice, the keyboard, and disk files. Output devices include most PC-compatible high-resolution bit-mapped CRT displays (e.g. IBM EGA, Tecmar Graphics Master, Hercules, Sigma 400). The system requires a computer equipped with a harddisk and at least 512 Kbytes of RAM.

Software

The IDS package is an interactive image display system which can be used to view digital images from remote sensing sources. In this regard, the operations comprising the system can be grouped into the following four classes: user interface, file manipulation operations, image enhancement routines, and output procedures.

i - User Interface

The system is designed for use by individuals who may have little or no experience with micro computer systems. Therefore, the user has been provided an user interface which is both easy to use and largely (!) foolproof. The former was achieved by implementing a menu format for the user interface, the latter through extensive error checking of user inputs from the keyboard.

The user interface consists of a bit-mapped display which is divided into several different static windows which contain the primary information displays for the system: the image itself, a histogram of the current image (or some subset), a text box displaying elementary statistics for the current image (or some subset), a title box, and two small windows which contain either co-ordinates or system status information.

Two levels of temporary ("pop-up") menus are available to the user. The main menu is quite large and intended as a quick reference to the available commands and need not be invoked to obtain the secondary menus. The secondary menus must be opened in order to use the more specialised functions (e.g. enhancement, filtering).

For short queries and responses a small "dialog box" at the bottom of the display is used. For more complex queries,

responses, and status/progress reports, a large window is temporarily opened. In all cases user keyboard input is carefully checked and entry errors trapped. Graphical input (i.e. pointing functions) may be performed with either the keyboard cursor keys or a mouse. At virtually any time during matrix operations or data input the user may abort the current step, restoring the system to its previous state.

ii - Files and Data Structures

The system was designed to read and write image data to and from disk. Consequently, a critical component of IDIPS is a set of file input/output operations. These operations allow the user to save a data file to disk after performing enhancement operations on that file, and to read a new data file into main memory from disk for display and/or manipulation. IDS is byte oriented and thus is designed to operate on images with pixel values in the range from 0 to 255.

iii - Enhancement and Filtering Operations.

The image processing routines consist of a small but comprehensive set of basic routines for the purpose of interactive image enhancement. The choice of routines was critical to the utility of the system, given the wide range of possible routines which could have been implemented. In addition, a set of statistical aids were included to aid the user in the assessment process.

The IDS package is designed for the interactive assessment of digital images. The operations which have been implemented are subdivided into the following classes:

- statistical aids
- image enhancement
- spatial filtering

For reasons of modularity, the latter two sets of operations have been allocated individual sub-menus. The statistical aids are executed from the main menu. The following chapters outline the various features which have been implemented and describe the algorithms employed (where appropriate).

Chapter II

Statistical Procedures

Digital images contain statistical information which is not visually obvious, but which may be highly useful in the assessment of image quality. The following aids have been incorporated in the IDS package.

- image histogram
- image parameters: minimum, maximum, mean,
mode, median, standard
deviation, number of pixels
- pixel extraction

The image histogram and image parameters are included as part of the main IDS display and are continuously present on the screen. Various keystroke combinations allow the user to update these displays for the entire image, or for any training rectangle which may be interactively specified. Pixel extraction enables the user to determine pixel frequencies, percentages, and class bounds for any user-specified rectangle.

Chapter III

Enhancement

Image enhancement involves the pixel by pixel transformation of digital images in order to improve visual discrimination of image components. This is achieved by transforming or 'stretching' the image data according to a specific mathematical function which is applied over the entire range of pixel values within an image, or to a subset thereof. The pixel by pixel nature of the transformation involved permits the image enhancement procedure to be implemented using an efficient table lookup algorithm.

Three image enhancement routines have been implemented in IDS; linear enhancement, piecewise enhancement, and automated enhancement. In each case a linear transformation function of the form below is employed. Out-of-range values are set to the end-of-range value. The general form of the transformation is:

$$T(i) = \frac{i - \text{low}}{\text{high} - \text{low}} * 255;$$

where i = pixel value
 low = low range delimiter
 high = high range delimiter

Linear contrast enhancement will enhance a single user specified range of pixel values over the range from 0 to 255. Piecewise contrast enhancement is designed to operate on images with asymmetric pixel distributions. The stretching operation is applied to two distinct ranges of pixel values thereby reducing histogram asymmetry and improving image contrast.

Automated contrast enhancement will saturate a user specified percentage of pixels high (255) and likewise low (0). The user is prompted for the percentage of pixels to saturate in each case, and the cumulative frequency distribution of the given image is used by the routine in order to determine the actual range of pixel values to stretch.

Chapter IV

Spatial Filters

Filtering techniques can be employed for purposes such as noise suppression, edge enhancement, or directional enhancement of image features. These techniques, like contrast enhancement, perform a pixel by pixel transformation on images. However, spatial filtering, unlike contrast manipulation, is context dependent in that the transformation applied to any pixel is dependent on the values of its neighbouring pixels. As a result, filtering techniques are more computationally intensive than are image enhancement techniques. To partially offset this computational burden an efficient box filter algorithm has been implemented for those filters where possible.

All of the filters included in IDS employ a 3 x 3 convolution matrix. The following filters have been implemented:

- smoothing - each pixel is set to mean of 8 neighbors
- median - each pixel is set to median of 8 neighbors
- mode - each pixel is set to mode of 9 pixels
- edge enhancement - each pixel is set to three times its own value minus the sum of non-diagonal neighbour pixels
- user defined - allows the user to specify his or her own convolution matrix.

The first three filters allow the user to smooth or remove random noise from an image. Edge enhancement acts to emphasize large gradients between adjacent pixels values thereby enhancing edges within an image. User defined filtering enables the user to apply any 3 x 3 convolution matrix which may be appropriate for specific applications.

Chapter V

Command Summary

Introduction

The following commands are available in the IDIPS system. They are presented in the order in which they appear in the various menus, i.e. in logical groupings rather alphabetically.

1 - Matrix Operations

Alt-L : Load Matrix

This loads a valid DOS file into one of the available bands. The system will request a valid file-name, and look for a header with image parameters appended to the file. If a header format is encountered with which the system is familiar, the the system will use this information to automatically load the data (if possible). Otherwise the user is prompted to manually enter the appropriate information (ie. number of lines and samples).

Alt-S : Save Matrix

This saves the current band-matrix into a file whose name is specified by the user. The user is prompted for a valid file-name.

Alt-H : Help

This opens a small window on the screen and presents the user with a branching menu of items upon which help is available. In fact, the help information is simply an edited version of this document, with branches to locations appropriate (??) to the user's needs.

Alt-Q : Quit

This allows exit of the system without saving either the parameter-record or the currently loaded image data.

2 - Rectangle Specification

Alt-T - Training Rectangles

A number of system operations (e.g. pixel extraction) require that the user select rectangular areas of the image to delimit either the area to be sampled, operated upon, or both. Pressing Alt-T initiates a procedure in which the user may interactively create, save, edit, and/or delete up to 8 different rectangles.

The user is prompted to select the training rectangle to be "edited". If it does not yet exist the user is prompted to specify, using a pointing device (keyboard or mouse), two opposite corners of the rectangle. The user may also change

a previously allocated rectangle by specifying its number, or delete any one by pressing <D> and giving a valid number at the prompt. One may also delete all of them by pressing <X> and confirming the request. One may also save them all to disk by pressing <S> and specifying a valid file-name (extension not required or necessary - '.RCT' automatically added).

Alt-J : Display Window

This allows the user to interactively specify the desired display rectangle within the current matrix dimensions. The user is prompted to use the pointer (mouse or keyboard) to select the opposite corners of the new display rectangle. When two corners have been selected, the system will automatically regenerate the main image window with the new display co-ordinates.

Note that if the specified rectangle is larger than the image itself (i.e. in the area where the axes and labels are drawn) the system will interpret the "oversize" specification as a request to increase the current rectangle size. This provides an easy method for expanding the display back to full dimensions after it has been "zoomed in" on some particular feature. To restore the display to full dimensions, simply specify the corners as the farthest lower-right corner and farthest upper-left corner.

3 - Update/Output Requests

Alt-G : Update All

This causes the system to re-calculate the statistics for the current band-matrix, update the main image, the elementary statistics window, the histogram and ogive.

Alt-I : Update Image

This causes the system to update the main image display, but does not cause it to update any of the statistics, either the calculations or the displays.

Alt-1..9 : Update Training Rectangle Statistics

This causes the system to calculate the current statistics for the specified training rectangle, n, (where <Alt-n> was pressed) and display them in the elementary statistics window and as a histogram and ogive.

Alt-0 : Update Matrix Statistics

This causes the system to re-calculate the statistics for the current data file, using the current operations rectangle.

Alt-= : Update Display Statistics

This causes the system to calculate and display the

statistics of the current data file based on the current display rectangle.

4 - Enhancement Menu

1 - Linear Stretch

This procedure uses user-specified lower and upper bounds to perform a linear stretch on the current image data over the indicated data range. The user is prompted for the two values and the enhancement is then performed. Values below the lower boundary are set equal to the lower boundary while value above the upper limit are set equal to the upper boundary.

2 - Piecewise Stretch

This procedure allows the user to apply differential linear stretching to different parts of the available bandwidth. The user is prompted for the number of ranges to be stretched, and is then prompted to enter the lower bound of each class.

3 - Automated Stretch

This procedure allows the user to specify the percentage of the observed frequency distribution to be saturated either low or high. The user is prompted for the percentage to be saturated low, and then the percentage to be saturated high. The routine uses the observed frequency distribution to find the data values corresponding to the specified degree of saturation, and then performs a linear stretch over that range.

5 - Filter Menu

1 - Smoothing (Moving Average)

This procedure uses a moving average (sum of all pixels in 3x3 cell divided by 9 placed in center cell) to smooth the pixel by pixel variation in the current image data. The user is prompted for the destination (band) of the result.

2 - Median

This procedure uses the median value of each 3x3 cell to replace the central value of each cell. The user is prompted for the destination (band) of the result

3 - Edge Detect

This filter enhances the spatial variation in pixel values, thus enhancing areas of rapid changes, or "edges". The filter operates by replacing the central value of the cell by the 8 times the central value less the sum of the other eight members of the cell. The user is prompted for the destination (band) of the result.

4 - Laplace

This procedure performs a convolution of the current image data using a 3x3 Laplacian filter, i.e. where the central value is replaced by 9 times the central value less the sum of the other eight values. The user is prompted for the destination of the result.

5 - User-Specified

This procedure allows the user to specify any values he wishes to be used as a mask/filter in a convolution over the current image data. The user is first prompted for the 9 values which make up the 3x3 cell, then the destination (band) of the result.

6 - Pixel Extraction Menu

1 - Point Sampling

This option allows the user to "roam" the present image with the cursor in order to examine, in detail, the distribution of values in the image. The user may exit the procedure by pressing <ESC>.

2 - Profile Sampling

This option the user to extract linear profiles along a specified transect. The user is prompted to move the cursor to the first endpoint of the profile and then press <CR>. As the cursor is moved to the second endpoint the system will sketch ("rubber band") the shape of the profile as the user moves the cursor. When the second endpoint is selected the system will redraw the entire final profile, draw a trace on

the image itself indicating the position of the transect, and then exit.

3 - Area Sampling

This option allows the user to select a rectangular area (either a previously allocated training rectangle or a user-specified rectangle) and extract summary statistics about the distribution of pixel classes. The classes used are those currently specified for the image, hence a maximum of eight. The user may, however, specify a different number of classes if he wishes. The user is initially prompted to specify a rectangle to be sampled, either the entire matrix, one of the training rectangles, or an interactively specified rectangle (option "W"). The resulting class histogram is plotted in the histogram window and the class statistics (text) are displayed in a window in the upper right. The area specified in the last column of the summary statistics are expressed in terms of "square pixels".

----- APPENDICES -----

Appendix A - System InstallationSystem Files

The IDS program consists of 4 main files:

IDIPS1.COM
IDIPS1.001
IDIPS1.002
IDIPS1.002

In addition, the system must have access to the three font files:

ROMANSIM.FNT
SYSTEM08.FNT
MARKERS.FNT

Also, the system must have access to the graphics driver software:

METAWNDO.EXE

All of these files must be in the directory from which IDIPS is started.

Setting Up the System

It is highly recommended that you load all of the IDS files and the font files into their own subdirectory of your hard disk and start the program from there. Two batch files have been provided, HDINSTAL.BAT and IDS.BAT. The former simply creates an \IDS subdirectory and loads all the files into it, and puts the IDS.BAT file in the root directory. The IDS.BAT file loads the MetaWindow driver and invokes IDS.